**Department of Electronic & Telecommunication Engineering**

**University of Moratuwa**

# BM2210 - Biomedical Device Design

# Final Report

**Group Members**

230218G GUNATHUNGA U.A.
230487D PERERA W.A.L.S.
230012U ABEYWARDHANA T.C.W.
230481E PERERA K.W.A.O.V.

November 30, 2025

# Contents

# 1. Introduction

## 1.1. Need Statement

### Key Problems Identified

1. Flow rates are manually set; deviations cause overdose/underdose risks.
2. Educated patients could help monitor flow if external displays exist.
3. Some patients self-adjust flows (dangerous for chemotherapy).
4. Regulators are expensive and rare in Sri Lanka.

**Population**

- Medical professionals– doctors, nurses, anesthetists, emergency staff.
- Patients– cancer, post-surgery, ICU, dehydrated, comatose/unresponsive patients

**Outcome**

Patients and medical professionals in the ward can adjust the IV fluid drip rate to the original set value by identifying possible deviations as indicated in the display.

**Final Need Statement**

There is a critical need for a safe, reliable, affordable, cost effective, doctor-friendly and patient-friendly method to externally regulate and display IV fluid drip rate in hospitalized patients.

## 1.2. Problem Status

The problem status can be explained with a user scenario as follows.

In a busy government hospital ward, a nurse or ward master is responsible for administering IV fluids to multiple patients at the same time. With traditional gravity-based infusion pumps, the ward master must manually set a timer and adjust the drip rate—often in increments of 3 drops per minute. However, patient movement or other disturbances can cause the flow to deviate, and since these pumps lack an external display, the ward master must physically check each patient every two hours to ensure the rate is still correct. This constant vigilance is time-consuming and stressful, and even then, deviations may go unnoticed.

In some cases, patients intentionally increase the drip rate—such as chemotherapy patients wanting treatment to finish early—creating further risk.

With our solution, the infusion pump displays the current drip rate in real time on an external screen. If the rate deviates from the user-set target, the system immediately alerts the ward master with an alarm. Even patients who try to adjust the rate themselves cannot exceed safe limits without triggering an alert. The ward master can monitor all patients' infusion rates simultaneously through a mobile app, reducing the need for constant physical checks. Additionally, if the rate drifts, the patient or nurse can manually adjust it using the control wheel while observing the display, ensuring the correct flow at all times.

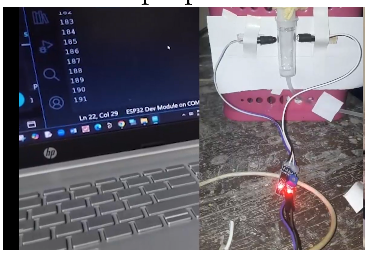
This approach improves patient safety, reduces the workload and stress on hospital staff, and brings the benefits of advanced infusion pumps to a system that is affordable and practical for government hospitals.

# 2. Concept Generation and Selection

## 2.1. Initial Concept Selection

We selected three initial concepts to move forward with:

1. Smart Gravity-Based Infusion Pump with IR-Based Drop Sensing
2. Smart Gravity-Based Infusion Pump with Capacitive Drop Sensing
3. Smart Gravity-Based Infusion Pump with Image Processing–Based Drop Sensing

| **Concept** | **Introduction** | **Testing Results** | **Advantages** | **Disadvantages** |
|---|---|---|---|---|
| IR-Based Drop Sensing | Basic principle of detecting droplets using infrared beam interruption. | Initial tests show consistent detection for clear and transparent liquids with a sensitivity of 1 drop per 200 ms.  | Simple setup, low cost, fast response time. | Affected by ambient light (sensitivity can be adjusted to avoid) |
| Capacitive Drop Sensing | Uses change in capacitance caused by presence of a droplet. | Initial testing showed that although the sensor responds to capacitance changes, obtaining clear and repeatable readings requires further calibration. The system also cannot reliably detect the exact moment a water droplet makes contact, and because the droplet volume is very small, the resulting capacitance change is minimal. Additional tuning and signal conditioning are therefore needed for accurate droplet-arrival detection. | High sensitivity for higher drop rates, works with transparent fluids. | Requires calibration, performance varies with humidity and temperature. |
| Image Processing–Based Drop Sensing | Detects droplets using camera images and computer vision algorithms. | Achieved high accuracy and detailed analysis in test runs. | Very accurate, provides size/shape information, flexible for many setups. | High computational cost, expensive hardware. |

### 2.2. Final Concept and Proposed Solution

**Final Concept:** Smart Gravity-Based Infusion Pump with IR-Based Drop Sensing

**Proposed Solution:** In our proposed solution, we aim to make infusion therapy more accessible and affordable by enhancing a simple, gravity-based infusion pump with the essential features listed below. Existing solutions are often inadequate for government hospitals, including specialized centers like cancer hospitals, due to the lack of external rate displays and monitoring—even in the most advanced infusion pumps currently used in the hospital system. Additionally, the high cost of current devices makes them unattainable for many healthcare facilities.

1. Mechanism to externally adjust the drip rate with real-time indication of drip rate in drops/min and mL (doctor-friendly and patient-friendly)

2. External display of drip rate in drops/min and mL

3. Alarm system for rate deviations

4. Mobile app/IoT-based monitoring system for real-time tracking across multiple beds in a ward

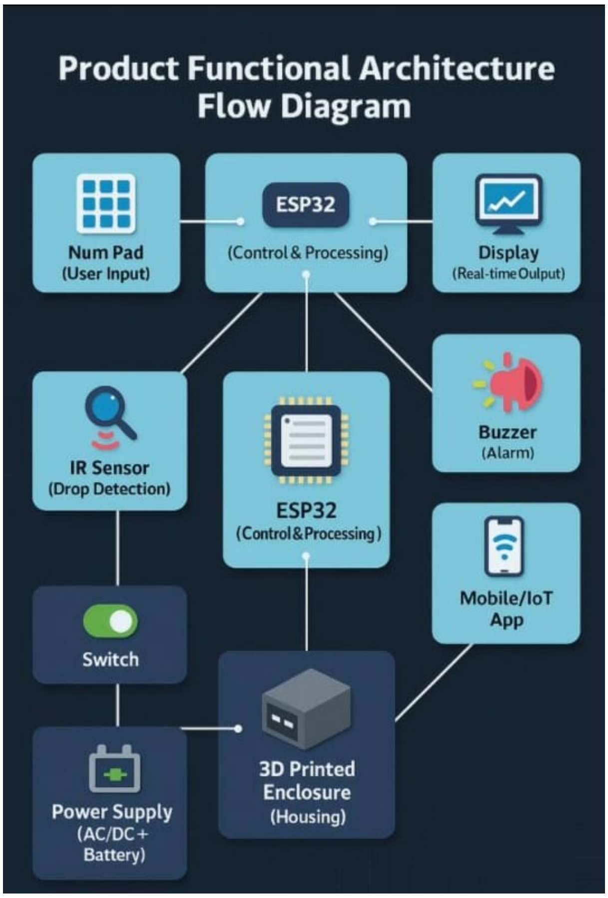
5. Mobile app/IoT-based monitoring system for real-time adjusting of the drip rate across multiple beds in a ward (for example: the ward master can remotely adjust the drip rate by identifying deviations from the originally set value)

## 3. Implementation Methodology of the Proposed Solution

### 3.1. Operating Mechanism of the IR-Based Drop Sensing

In this method, a standard infrared (IR) sensor module is used and rewired, so that the IR emitter and receiver are placed directly opposite each other across the IV drip chamber just before the fluid enters the tube. When no drop is present, the IR beam passes interrupted and is detected continuously. Each falling drop interrupts the beam, producing a momentary signal change that is captured by the photodetector. These interruptions are counted by an ESP32 microcontroller, which converts the drop count into clinically relevant infusion rates (drops/min) and displays on a seven-segment display. This design requiresonly three core components, ESP32 module, IR module, and OLED screen that makes this method simple, low-cost, and highly reliable and ideal for affordable hospital use.

## 3.2. Functional Block Diagram

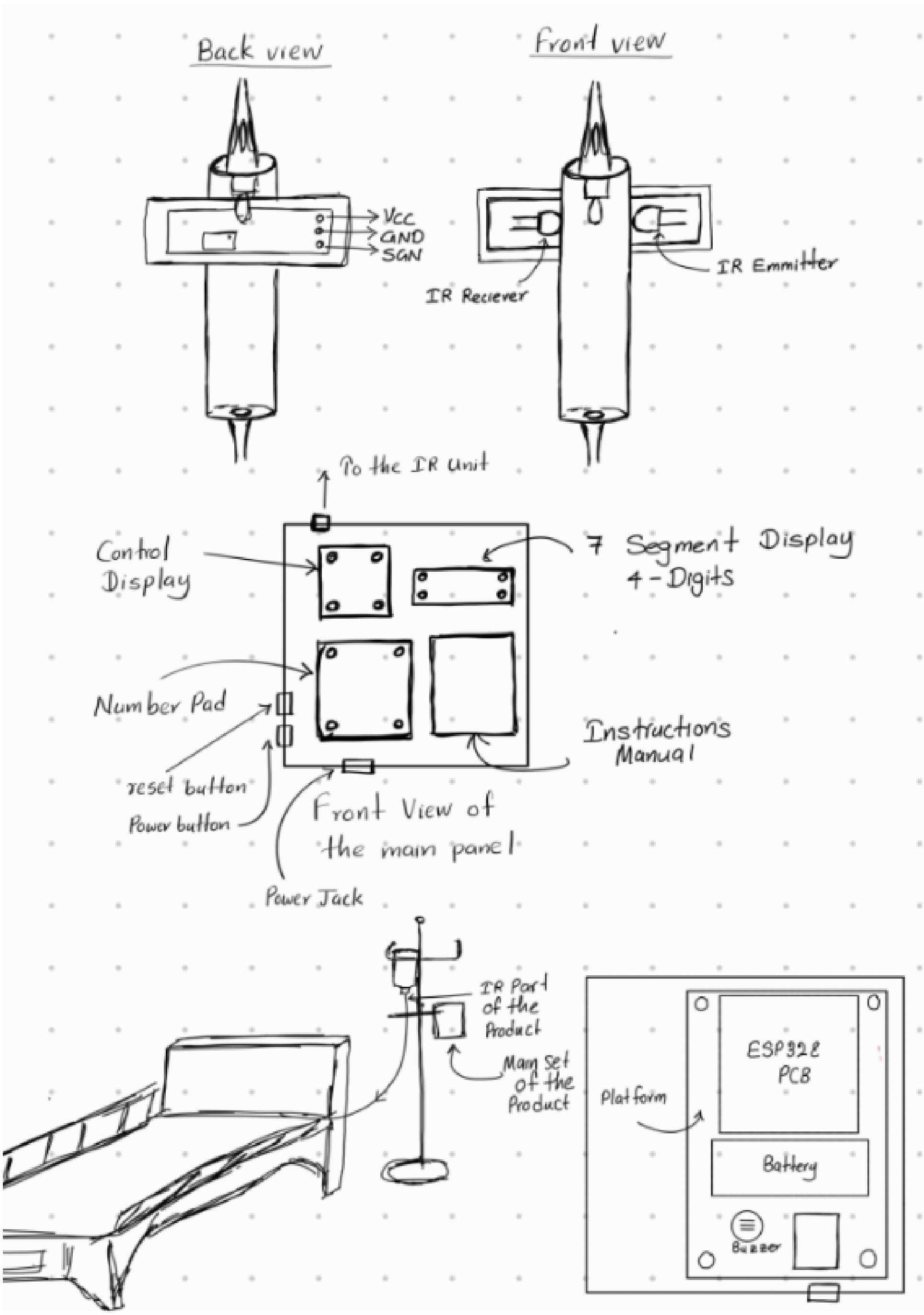



## 3.3. Component Selection

| Component | Role in the Product |
|---|---|
| ESP32 | The brain of the system. Processes signals from the IR sensor, calculates drop rate, displays results, and connects with mobile/IoT systems. |
| Buzzer | Acts as an alarm. Alerts staff immediately when the drip rate deviates from the expected value. |
| Num Pad | Allows medical staff to input the desired drip rate (drops/min) for each patient. |
| Display | Shows the real-time drip rate in drops/min and mL, making it doctor-friendly and patient-friendly. |
| TP4056 | Manages battery charging with built-in overcharge protection, ensuring safe and reliable backup power. |
| DC Jack | Provides AC power connection for continuous operation in the ward. |
| IR Sensor | Detects each IV fluid drop when it interrupts the light beam, enabling accurate real-time drop counting. |

## 3.4. 3D Enclosure Design

### 3.4.1 Initial Sketch



### 3.4.2 Evolution of the 3D Design

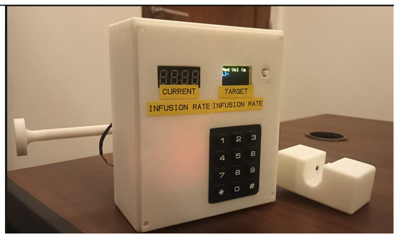
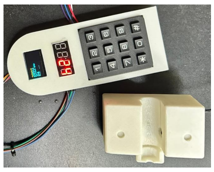
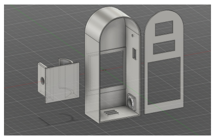
| |   **Initial Design** |   **Final Design** |
|---|---|---|
| **Overall Dimensions** | Larger footprint, occupies more volume | Significantly smaller footprint, optimized volume |
| **Internal Component Arrangement** | Relaxed internal spacing, easy routing of wiring and PCBs | Tighter internal layout, requires careful component placement |
| **Portability** | Relatively heavy and bulky, not convenient for frequent relocation | Lightweight and highly portable, easier bedside handling |
| **Ease of Assembly & Maintenance** | Good component access, straightforward assembly and servicing | Moderate compact layout reduces accessibility and requires planned service access points |
| **Material Usage** | Higher material consumption (larger panels and supports) | Reduced material usage leading to lower per-unit material cost |
| **Aesthetic & Ergonomics** | Boxy and functional, less bedside friendly aesthetic | Sleeker, modern appearance, better suited to clinical bedside environments |
| **Mounting Options** | Requires a larger mounting surface or custom bracket | Easier to mount on IV poles, bed frames, or compact bedside brackets |
| **Stability & Rigidity** | Inherently stable due to larger base and mass | Needs optimized structural ribs/walls to retain rigidity in a smaller form factor |
| **Clinical Usability** | Buttons and labels can be spaced for clarity and to prevent accidental presses | Controls are compacted layout must avoid accidental activation and remain legible |
| **Battery & Electronics Capacity** | Room for a larger battery pack and extra modules for expansion | Limited internal volume constrains battery size and future addons |
| **Manufacturability** | Simpler tooling for assembly, but higher material cost | May require more precise tooling and inserts; lower material cost per unit |
| **Weight** | Higher overall weight (less mobile) | Lower weight improves portability and reduces shipping costs |
| **Cable Management** | Ample space for routing, strain relief, and connectors | Limited routing channels; requires careful cable planning and retention features |

Table 1: Comparison between Initial and Final 3D enclosure designs.

## 3.5. IoT-Enabled Web Platform for IV Fluid Monitoring

To enhance the functionality, accessibility, and reliability of our IV fluid drop-rate monitoring device, we integrated it with an IoT-based web platform. This online system enables real-time monitoring of infusion status, patient information, and alert conditions from any location within the hospital network. The website acts as the central communication hub between the hardware device and the medical staff, ensuring continuous and accurate data visibility.

**System Architecture Overview**

The web platform is built using a modern, modular IoT architecture consisting of a Python backend, a React.js frontend, an SQL database, and REST-based API communication. Each component was selected to ensure reliability, scalability, and smooth real-time data handling.

### 3.5.1 Backend

The backend of the system is implemented using Python due to its strong ecosystem for IoT and data processing, high reliability for handling sensor data from the ESP module, seamless integration with databases and APIs, and fast development and debugging capabilities—crucial for rapid prototyping.

### 3.5.2 Frontend

The frontend website was developed using React.js, chosen for its ability to deliver fast, dynamic, real-time interfaces. It supports component-based development for modularity and maintainability. React's efficient state management ensures smooth live updates of drop rate, warnings, and patient data without requiring page reloads.

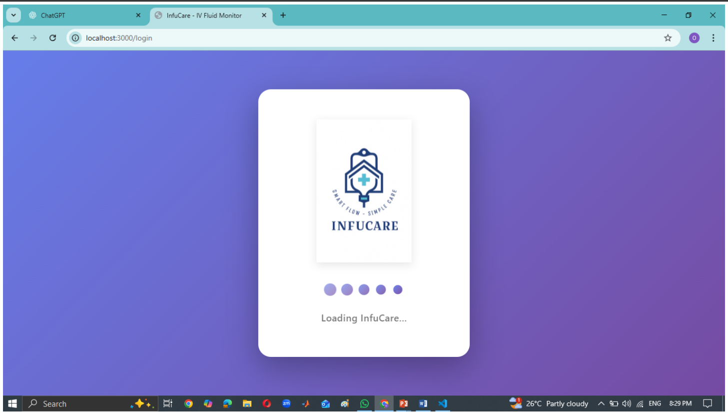
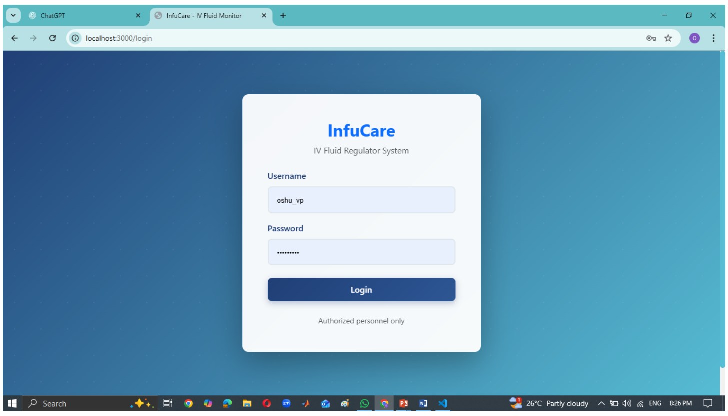
**Purpose of the Web Platform**

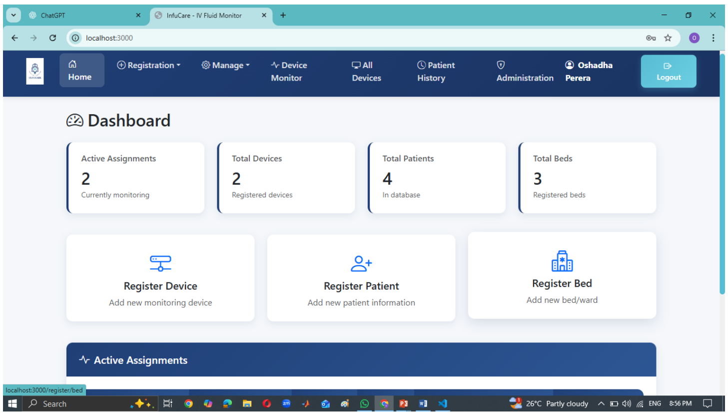
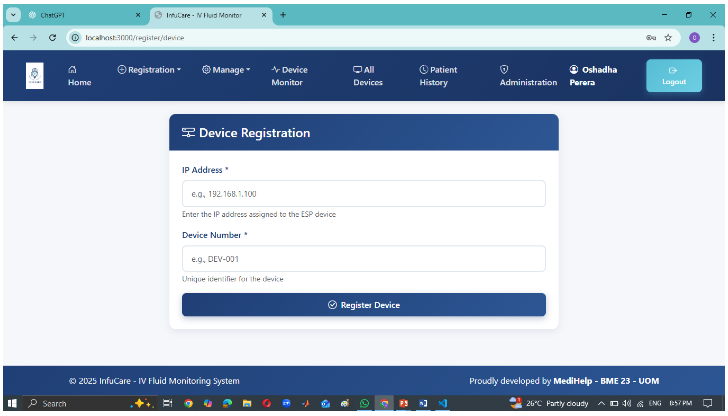
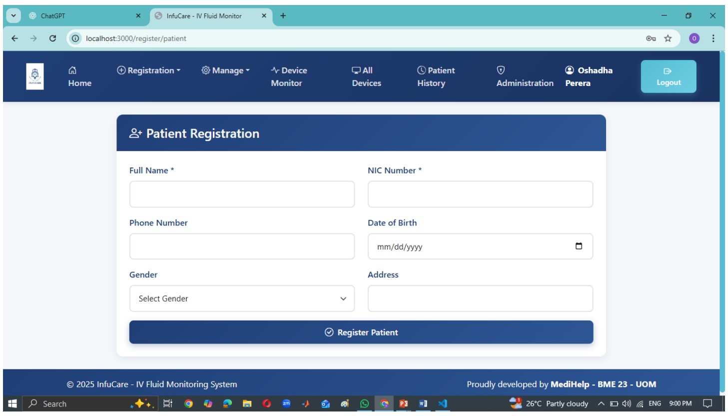
The purpose of developing a dedicated website for monitoring the IV fluid drop-rate device includes the following:

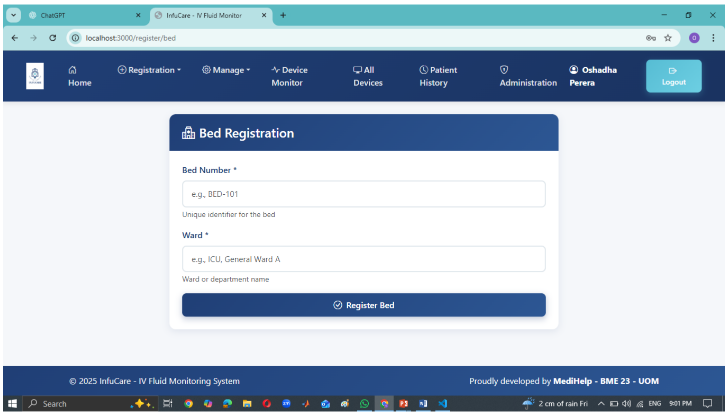
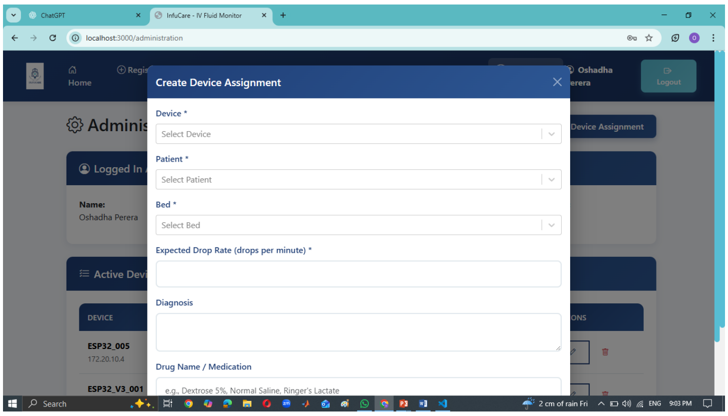
1. **Real-Time Monitoring:** Allows nurses to view the live drop rate and infusion status from any workstation without physically approaching the drip stand.
2. **Early Warning and Alerts:** If the drop rate deviates, slows, or stops, the website displays warning and critical alerts. This enables immediate medical action and reduces infusion errors.
3. **Remote Accessibility:** Staff can monitor multiple patients across different wards through one centralized dashboard.
4. **Data Logging and History:** All infusion sessions are stored, assisting with patient documentation, medical audits, and analysis of infusion trends and device performance.
5. **Improved Workflow Efficiency:** Reduces the need for manual drop counting, allowing medical staff to focus on more critical care activities.

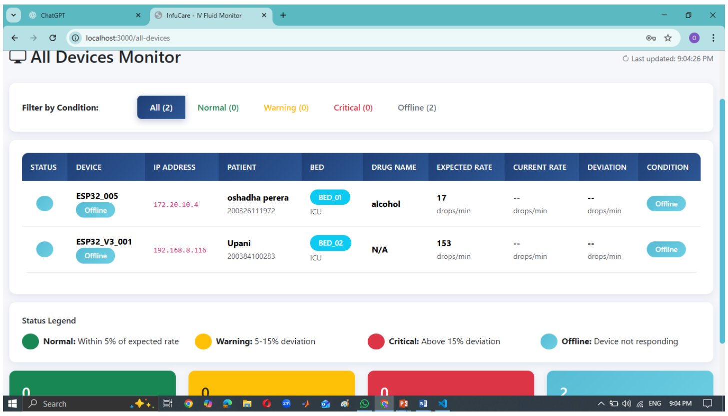
6. **Scalability:** More devices can be added easily due to the API-based modular system design.

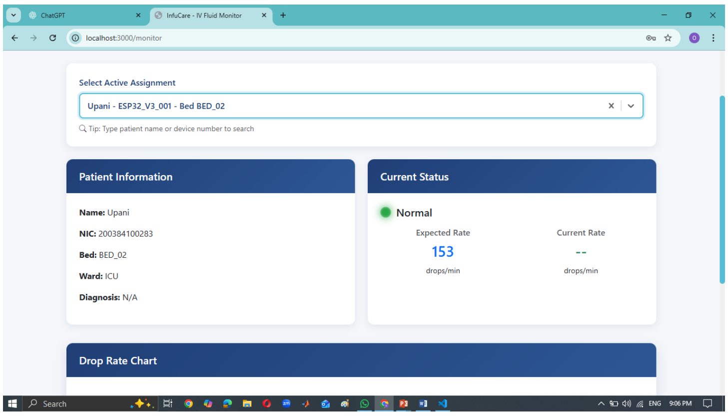
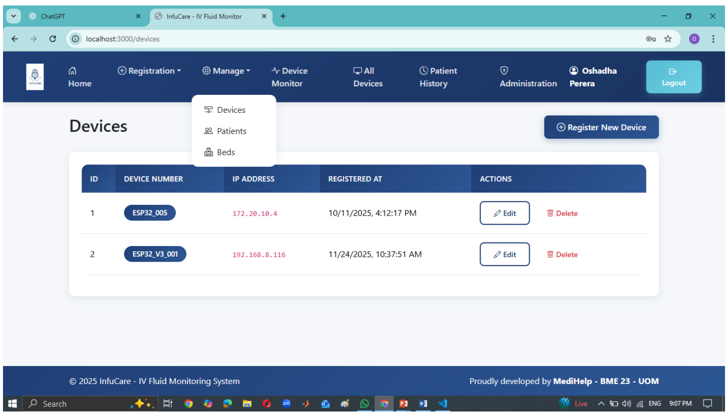
## 3.6. User Interface

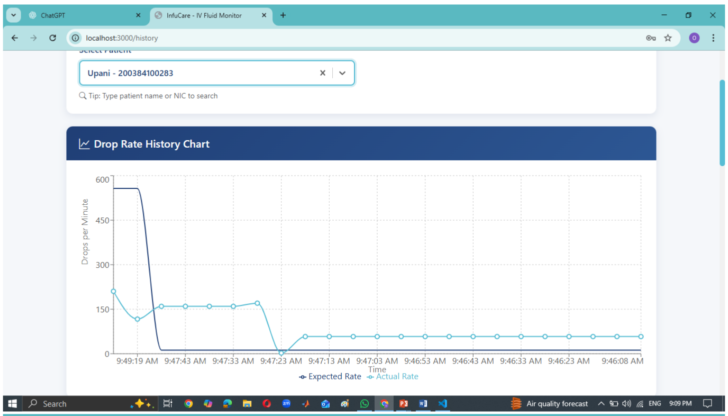
| Page Name | Image | Description |
|---|---|---|
| Load Page |   | The website is designed with a simple, clear, and secure interface to ensure smooth use by hospital staff. Each section of the interface supports a specific task in the IV fluid monitoring workflow, starting from logging in to monitoring patients and managing historical data. |
| Login Interface |   | The hospital administration begins by logging into the website through a secure login page. Password protection ensures that only authorized staff can access the system. This protects patient information, maintains privacy, and prevents unauthorized changes to infusion details or device settings. Security is essential in a hospital environment, and this interface ensures that all data is accessed only by verified users. |

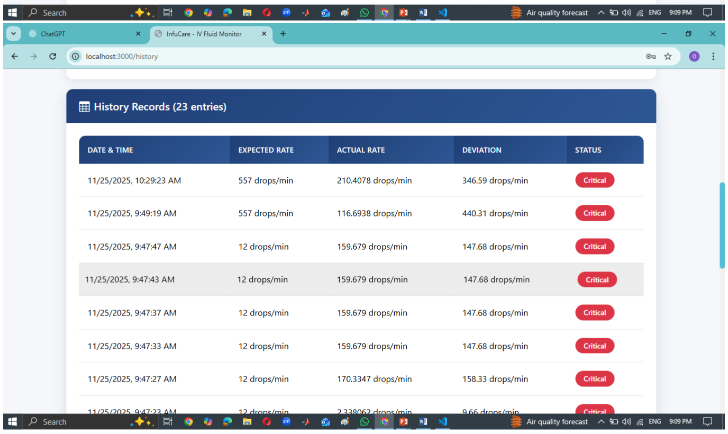
<table>
<tr><th>Page Name</th><th>Image</th><th>Description</th></tr>
<tr><td>Home Page Dashboard</td><td>
</td><td>The Home Page provides a quick overview of the system. It shows the total number of active assignments, total devices registered, total patients, and total beds in the system. From here, the admin can register a new device, register a patient, or register a bed to a specific ward. It also displays all ongoing active assignments, giving a clear snapshot of the current workload and device usage across the hospital.</td></tr>
<tr><td>Device Registration Interface</td><td>
</td><td>The Device Registration page is used when a new device is introduced to the system for the first time. The admin must enter the device's IP address and device number both of which are unique identifiers. This ensures that each device is mapped accurately and prevents conflicts when reading drop-rate data.</td></tr>
<tr><td>Patient Registration Interface</td><td>
</td><td>The Patient Registration interface allows the admin to enter and save patient details into the system. The required information includes the patient's full name, NIC number, and other relevant details. Registering the patient ensures that all infusion data is correctly linked to the correct individual in the hospital.</td></tr>
</table>

| Page Name | Image | Description |
|---|---|---|
| Bed Registration Interface |  | The Bed Registration page lets the admin assign a bed to a ward. A new bed can be added, or an existing bed can be reassigned to a different ward. The bed number and corresponding ward must be entered. This helps maintain an accurate map of where each patient and device is physically located. |
| Administration Interface – Assignment Creation and Editing |  | After all basic setups are complete, the Administration interface is used to create new infusion assignments. Here, a bed, a patient, and a device are linked together. The admin also enters the required drop rate, the drug name being infused, and the diagnosis. This page also allows editing an assignment which is useful when a previously assigned device needs to be moved to another patient or when patient details change. |

<table>
<tr><th>Page Name</th><th>Image</th><th>Description</th></tr>
<tr><td>All Devices Interface – Main Monitoring Page</td><td>
</td><td>The All Devices page is the core of the system. It displays every device along with its current drop rate, required drop rate, and any deviations. The page shows whether a device is online or offline, its IP address, and the drug being administered. Warning and critical alerts are shown clearly, helping staff identify problems immediately. This is the most important page, as it provides the complete real-time monitoring overview for all active infusions.</td></tr>
</table>

| Page Name | Image | Description |
|---|---|---|
| Device Monitor – Individual Device View |  | The Device Monitor interface allows users to search by device number, patient name, or bed number to view one specific ongoing assignment. It shows full patient details along with the live drip count and deviation indicators. Percentage = Deviation x 100% / required drop rate Normal (Green): Percentage ¡ 5% Warning (Yellow): Percentage ¡ 15% Critical (Red): Percentage ¿ 15% for 5 consecutive readings At the bottom, a real-time graph displays the drop rate versus time. The required drop rate is shown as a straight green line, while the live drop rate is plotted dynamically, allowing staff to quickly understand infusion stability and trends. |
| Settings (Manage) Interface |  | The Settings page provides full administrative control. Staff can edit device information, edit patient details, or update bed records. It also includes the option to delete any unnecessary or outdated information, helping maintain a clean and accurate database. |

<table>
<tr><th>Page Name</th><th>Image</th><th>Description</th></tr>
<tr><td>Patient History Interface</td><td>




</td><td>The Patient History interface allows staff to search for a specific patient and view all previous infusion records. The data is shown both as a graph and in a grid format, including recorded conditions such as normal, warning, or critical states. An important design choice is that the system stores data at 1 minute intervals. Since the device sends continuous real-time readings, saving every single reading would overload the database and cause storage issues. This optimized method keeps the system fast, efficient, and reliable while preserving meaningful historical trends.</td></tr>
<tr><td>Log Out Option</td><td></td><td>Finally, a simple logout button allows users to securely exit the system. This helps maintain privacy and ensures that only authorized staff continue to have access.</td></tr>
</table>

## 3.7. Final Product and Product Dissection

A PCB was not printed and hardware components were soldered on to a dot board.

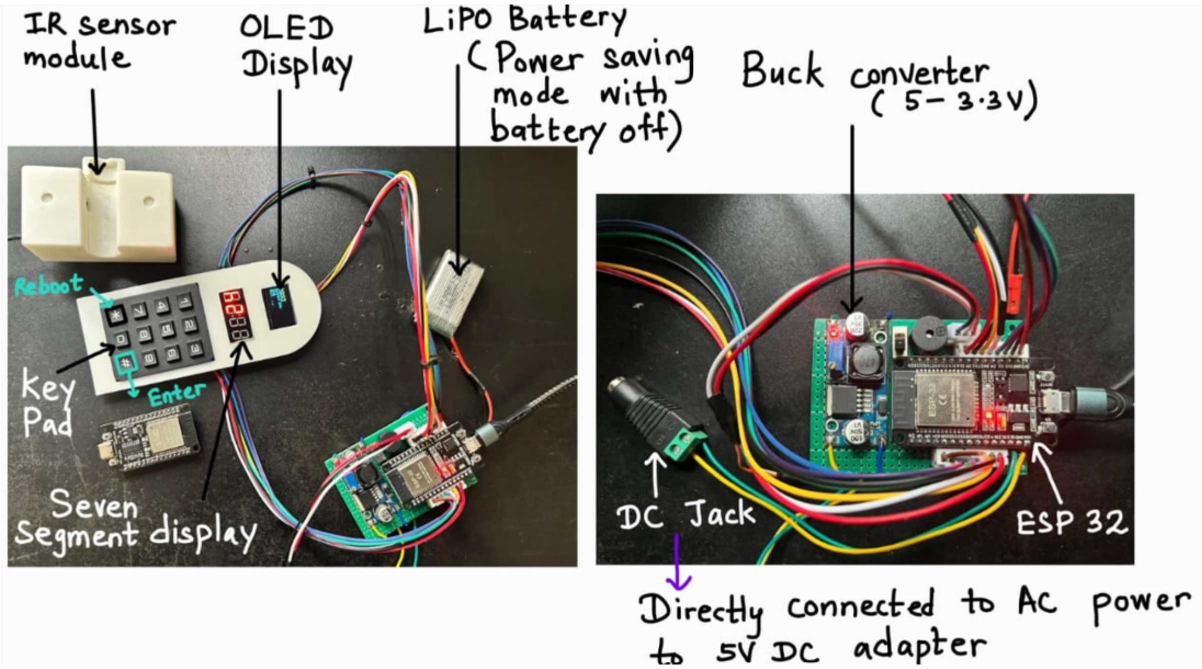



Figure 1: Product Dissection


Figure 2: Final Product

# 4. Accuracy Testing Results

| **Name of Test** | **Objective** | **Procedure** | **Pass / Fail Criteria** | **Status** |
|---|---|---|---|---|
| Volumetric Chronometric Accuracy Test | Check if measured drop rate matches actual infusion rate | Set target volume and infusion time on the device. Measure drops with both device and manual reference (stopwatch & count). Repeat at low, medium, and high flow rates. | Pass: Device reading within ±3% or ±2 drops/min of reference at all flow rates.<br>Fail: Deviation beyond this limit in any test condition. | Pass |
| Photonic Sensitivity Validation | Verify IR sensor detects all drops correctly | Generate controlled drops using a calibrated IV set. Record counts with device and independent reference . Test under different lighting conditions. | Pass: more than 99% drops detected, less than 1% false positives.<br>Fail: Any frequent misdetection, missed drops, or high false alarms. | Pass |
| Alarm Responsiveness Verification | Confirm alarms trigger for deviations | Set required drop rate on the device. Introduce slow flow (20%), fast flow (+20%), and blockage (0 drops). Record buzzer, LED, and web-app alarm response time. | Pass: Alarm activates immediately after the deviation.<br>Fail: Alarm is delayed, missing, or activates during normal operation. | Pass |
| Long-Duration Stability Test | Ensure accuracy over extended operation | Run the device continuously for 24 hrs for a 500 ml saline volume infusion. Compare readings with reference every 6 hrs. Monitor for drift and error accumulation. | Pass: Cumulative drift less than ±2% over 24 hrs and device remains functional without resets.<br>Fail: Drift greater than ±2%, frequent errors, or device malfunctions. | Pass |
| Power Continuity and Recovery Test | Test battery backup and recovery after power loss | Switch from AC to battery power during normal operation. Measure backup run-time until battery depletion. Restore AC power and check for data retention. | Pass: Backup lasts for more than 8 hrs, data/settings remain intact, operation resumes within 10 s.<br>Fail: Backup less than 8 hrs, data lost/corrupted, or system fails to resume properly. | In Progress |
| Data Transmission Integrity Test | Ensure correct web app data logging | Connect device to network and start continuous transmission. Introduce unstable conditions (packet loss, disconnects). Compare hospital database logs with device logs. | Pass: Data delay less than 10 s, recovery after disconnection, no missing records.<br>Fail: Frequent packet loss, incorrect timestamps, or missing records. | Pending |

| | | | | |
|---|---|---|---|---|
| User Interaction Validation | Confirm ease of use and prevention of input errors | Input volume/time and start infusion. Test functions: alarm mute, acknowledge, and calibration. Enter incorrect values to test error detection. | Pass: more than 90% of tasks completed correctly; input errors such as unrealistic volume are flagged and prevented.<br>Fail: Frequent operator mistakes without system warning, unclear interface, or failure to accept correct input. | Pass |

# 5. Final Budget and Pricing Strategy

Final budget was calculated by adding the cost of PCB printing.

| Expense | Unit price (Rs.) | Quantity | Amount (Rs.) |
|---|---|---|---|
| ESP32 WiFi Bluetooth Dev Board | 1390 | 1 | 1390 |
| 3x4 Matrix Keypad Module | 840 | 1 | 840 |
| 7 Segment 4 Bits Digital LED Display Module | 345 | 1 | 345 |
| OLED Display 0.96" I2C | 690 | 1 | 690 |
| IR Infrared Obstacle Avoidance Sensor Module | 125 | 1 | 125 |
| AC DC Power adopter 5 V | 590 | 1 | 590 |
| DC power socket plug jack connector | 60 | 1 | 60 |
| 6x2mm Neodymium Magnet | 30 | 4 | 120 |
| Buzzer | 50 | 1 | 50 |
| TP4056 charging module | 50 | 1 | 50 |
| Switches | 190 | 2 | 380 |
| Miscellaneou (wires etc.) | | | 150 |
| Component Cost | | | 4790 |
| PCB print | 2980 | 1 | 2980 |
| Enclosure Print | 3650 | 1 | 3650 |
| TOTAL COST | | | 11420 |

Figure 3: Budget for production of one item

Cost Per Unit ( Bulk Production) +100 Units

| Expenses | | Amount per Item |
|---|---|---|
| Component Cost ( 30-50% decrement)- 40% off | 4790 x 0.60 | 2874 |
| Batch 3D printing (+100) | 800-1400 | 1100 |
| PCB Print (+100) | 300 - 600 | 450 |
| Manufacturing and Assembly cost (+100) (soldering, wiring etc.) | 400- 600 | 500 |
| Packaging and Logistics (Box, Cushioning, labelling etc.) | 300 -500 | 400 |
| Cost per unit | | 5324 |
| Software Implementation - One time Software Licence Fee | | 1500 |
| Marketing Sales & Operational overhead (10-20%) - 15% | 5324x 0.15 | 799 |
| Profit Margin (15-40%) - 20% Profit | (5324 + 799 ) x 0.2 | 1225 |
| Estimated Total Price per unit | | 8850 |

TOTAL UNIT PRICE = Rs. 8850 .00

Figure 4: Budget for bulk production of 100 units

# 6. Possible Regulatory Pathway

| **Regulatory Pathway** | **Nature / Required Submissions** |
|---|---|
| Device Classification | Likely classified as a FDA Class II device. The device is non-invasive and is required to perform as expected and will not cause injury or harm to the user. |
| Ethical Approval | **Objective:** Needed to conduct initial clinical trials.<br>**Required Submissions:** Submit documents to obtain ethical clearance from the University Ethics Review Committee, University of Moratuwa (`https://uom.lk/uerc`). |
| Preclinical Testing | **Objective:** Validate functionality, safety, and usability.<br>**Required Submissions:**<br>• Performance Testing: Validate functionality<br>• Electrical Safety and EMC: Test compliance with IEC 60601-1 and IEC 60601-1-2<br>• Usability: Conduct human factors studies per FDA guidelines |
| NMRA Approval | **Objective:** To conduct clinical trials.<br>**Required Submissions:** Submit documents to obtain clearance from the National Medicines Regulatory Authority (NMRA), the responsible body for the regulation and control of registration, licensing, manufacture, importation and all other aspects pertaining to medical devices in Sri Lanka to further continue clinical trials. |
| Software and Cybersecurity | **Objective:** Comply with IEC 62304 for software development and include testing results in the submission. Address cybersecurity risks with FDA's cybersecurity guidance, such as data encryption and risk mitigation. |
| Premarket Submission (510(k)) | **Required Submissions:**<br>• Device description and intended use<br>• Performance and safety test results<br>• Risk analysis per ISO 14971<br>• Labeling |
| Manufacturing and Post-market Compliance | **Objective / Requirements:**<br>• Implement a QMS compliant with 21 CFR Part 820 (Quality System Regulation)<br>• Register the device and manufacturing facility with the FDA<br>• Monitor post-market performance and report adverse events via the Medical Device Reporting (MDR) system |

# 7. Conclusions and Future Improvements

### 7.0.1 Conclusion

The key uniqueness of our solution is adding an external display of the current infusion rate (drops per minute) to the simplest gravity-based infusion pumps commonly used in hospitals which is even a lacking feature in advanced pumps currently used in the hospital system as rate can change with patient movements. Our system provides alarms whenever the infusion rate deviates from the user-set target, ensuring patient safety without requiring expensive advanced pumps that can cost around Rs. 100 000.00. This hybrid approach delivers the core benefits of advanced infusion pumps—real-time monitoring and deviation alerts—in an affordable, user-friendly package suitable for everyday hospital use.

Advanced IV fluid regulation systems are not affordable to the local hospital system. At the moment, no one is providing a low-cost but reliable IV regulator made for the Sri Lankan context. This creates a clear space in the market that our solution can fill.

In conclusion, our team approached the development of this infusion pump with careful attention to both component selection and overall design. Every decision was made with the goal of meeting the exact functional requirements while ensuring reliability, safety, and usability. To achieve this, we consciously prioritized essential features, streamlining the device by omitting additional "smart" functions that, while interesting, were not critical to its core operation. This approach allowed us to create a solution that is not only cost-effective but also user-friendly and accessible, ensuring that it can serve its purpose efficiently in a real-world clinical setting.

### 7.0.2 Future Improvements

1. The system will be enhanced by enabling ESP32-to-ESP32 communication within each hospital ward. This will allow the creation of a local LED indicator network connecting all infusion setups in the ward. Each bed will have a dedicated LED indicator that reflects the current drip status — normal, deviating, or alarm — in real time.

   This offline communication model will ensure that the monitoring and alert functions remain fully operational even if internet connectivity fails, making the system highly suitable for rural hospitals and low-resource environments where reliable internet access cannot be guaranteed.

2. We plan to keep the battery attachment and BMS as future improvements, while for now, we are focusing on implementing a power-saving mode to extend battery life. This mode will selectively turn off high-power components, such as the Wi-Fi module, when they are not needed, reducing energy consumption. By combining this with optimized microcontroller sleep states and minimal sensor activity during idle periods, we aim to maximize the efficiency of the current system while leaving room for future enhancements like advanced battery monitoring and management.

3. We are planning to add a software-based lock mechanism to the keypad to prevent patients from changing the drop rate, ensuring safety and consistent operation. Additionally, we are enhancing the system's security in the web application by implementing two-step verification, providing an extra layer of protection for user

accounts and sensitive data.

**For the web application,**

4. One major improvement is reducing the amount of data the staff has to enter manually. In our current system, nurses and administrators must re-enter patient details, device information, drug data, and other fields specifically for IV infusion monitoring. But in a real hospital environment, this information is already recorded at the time of patient admission through the hospital's existing management software (HMS/EMR systems). Re-entering all this data again only for infusion monitoring is not practical and increases workload.

5. Another future enhancement is to make our web application work as an add-on module that can communicate directly with the hospital's existing software using APIs or standardized healthcare data exchange protocols. With this integration, once a patient is admitted in the hospital system, all relevant information can automatically sync to our device-assignment interface. This allows staff to assign devices quickly without typing the same details again. Alternatively, the infusion assignment feature itself can be embedded as a new function inside the hospital's existing software. This makes the system more realistic, efficient, and ready for real-world deployment in a busy clinical environment.

# 8. Task Allocation

| Name | Index Number | Contribution |
|---|---|---|
| GUNATHUNGA U.A. | 230218G | • Initial feasibility check of IR drop sensing method<br>• Developing the IR sensing module with required initial functionality and its 3D Design<br>• Content creation for project pitching and documentation |
| PERERA W.A.L.S. | 230487D | • Initial feasibility check of image processing method for drop sensing<br>• Component selection<br>• Microcontroller programming<br>• Soldering and assembling<br>• Documentation |
| ABEYWARDHANA T.C.W. | 230012U | • Initial feasibility check of capacitive method for drop sensing<br>• Initial sketch of 3D design for part of the device fitting to the pole<br>• 3D design development using Fusion<br>• Documentation |
| PERERA K.W.A.O.V. | 230481E | • Developing the web application<br>• IoT integration of the device<br>• Soldering and assembling<br>• Documentation |

# 9. Program for ESP32 Microcontroller

```
#include <Keypad.h>
#include <TM1637Display.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include "esp_system.h"
#include <WiFi.h>
#include <WebServer.h>
#include <ArduinoJson.h>

const byte ROWS = 3;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1','4','7','*'},
  {'2','5','8','0'},
  {'3','6','9','#'}
};

byte rowPins[ROWS] = {27, 26, 25};
byte colPins[COLS] = {33, 32, 14, 12};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

#define CLK 2
#define DIO 15
TM1637Display display(CLK, DIO);

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SDA_PIN 21
#define SCL_PIN 22
#define BUZZER_PIN 13
Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// ========== WiFi CONFIGURATION ==========
const char* ssid = "Dialog 4G - Indeepa"; // Change this to your WiFi name
const char* password = "Indeepa@05"; // Change this to your WiFi password
String deviceNumber = "ESP32_V3_001"; // Unique device identifier
String deviceIP = "";
WebServer server(80);

#define DROP_PIN 4
volatile unsigned long dropCount = 0;
volatile unsigned long lastDropTime = 0; // microseconds

// new alarm stages
const unsigned long deviationDelay = 5000; // 5 seconds
const unsigned long patternDuration = 3000; // 3 seconds

bool patternPhase = false;
bool continuousPhase = false;

unsigned long melodyPrev = 0;
unsigned long panicPrev = 0;

bool inMelody = false;
bool inPanic = false;

int melodyStep = 0;
int panicStep = 0;

unsigned long medicineVolume = 0; // mL
unsigned long durationMinutes = 0;
float presetDropRate = 0; // DPM
unsigned long starttime = 0;
unsigned long alarmBlockUntil = 0; // Track start time for runtime calculations

// buzz blinking
unsigned long previousBlinkTime = 0;
bool buzzState = false;
const unsigned long blinkInterval = 250; // ms

// display updating
unsigned long previousDisplayMillis = 0;
const unsigned long displayInterval = 500;

// Beeping overtime correction
unsigned long overLimitStart = 0;
bool alarmActive = false;
const unsigned long overLimitDelay = 10000;

#define NUM_SAMPLES 5
volatile unsigned long dropTimes[NUM_SAMPLES];
volatile int dropIndex = 0;
bool enoughDrops = false;

float currentDPM = 0.0;

// ========== FUNCTION DECLARATIONS ==========
void connectWiFi();
void setupWebServer();
void handleGetData();
void handleRoot();
void handleNotFound();

// ISR drop detection
void IRAM_ATTR onDropDetected() {
    unsigned long now = micros();

    if (now - lastDropTime > 150000) {
        dropTimes[dropIndex] = now;
        dropIndex = (dropIndex + 1) % NUM_SAMPLES;
        dropCount++;
        lastDropTime = now;

        if (dropCount >= NUM_SAMPLES) enoughDrops = true; // start DPM calculation
    }
}

// keyboard input
unsigned long getUserInput(String prompt) {
    String input = "";
    oled.clearDisplay();
    oled.setTextSize(2);
    oled.setTextColor(SSD1306_WHITE);
    oled.setCursor(0,0);
    oled.println(prompt);
    oled.display();

    while (true) {
        server.handleClient(); // Handle web requests during input
        char key = keypad.getKey();
        if (key) {
            if (key >= '0' && key <= '9') {
                input += key;
            } else if (key == '*') { // backspace
                if (input.length() > 0) input.remove(input.length() - 1);
            } else if (key == '#') { // confirm
                if (input.length() > 0) return input.toInt();
            }

            // Display current input
            oled.clearDisplay();
            oled.setTextSize(2);
            oled.setCursor(0,0);
            oled.println(prompt);
            oled.setTextSize(3);
            oled.setCursor(0,30);
            oled.println(input);
            oled.display();
        }
    }
}

// --- Calculate instantaneous DPM using last few drops ---
float calculateInstantDPM() {
    if (!enoughDrops) return 0; // wait until 5 drops detected

    // Find oldest and newest drop in buffer
    unsigned long oldest = dropTimes[dropIndex]; // circular buffer
    unsigned long newest = dropTimes[(dropIndex + NUM_SAMPLES - 1) % NUM_SAMPLES];

    unsigned long dt = newest - oldest; // total microseconds between 5 drops
    if (dt == 0) return 0;

    float dpm = (NUM_SAMPLES - 1) * 60.0 * 1000000.0 / dt; // extrapolate to drops per minute
    return dpm;
}

void playMelodyPattern() {
    static int sequence[] = {1,0,1,0,0,1,1,0}; // melody feel
    static int len = 8;

    if (millis() - melodyPrev > 200) { // slow beautiful
        melodyPrev = millis();
        melodyStep = (melodyStep + 1) % len;
        digitalWrite(BUZZER_PIN, sequence[melodyStep]);
    }
}

void playPanicPattern() {
    const unsigned long interval = 80; // constant fast beeping without overload
    if (millis() - panicPrev > interval) {
        panicPrev = millis();
        digitalWrite(BUZZER_PIN, !digitalRead(BUZZER_PIN));
    }
}


void setup() {
    Serial.begin(115200);
    pinMode(BUZZER_PIN, OUTPUT);
    digitalWrite(BUZZER_PIN, LOW);

```

```
    // --- TM1637 ---
    display.setBrightness(0x0f);
    display.clear();

    // --- OLED ---
    Wire.begin(SDA_PIN, SCL_PIN);
    if(!oled.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println("OLED init failed");
        for(;;);
    }
    oled.clearDisplay();
    oled.display();
    alarmBlockUntil = millis() + 10000; // 5 minutes

    // Welcome screen
    oled.clearDisplay();
    oled.setTextSize(1);
    oled.setTextColor(SSD1306_WHITE);
    oled.setCursor(0,0);
    oled.println("--------------------");
    oled.setCursor(0,10);
    oled.println(" Team InfuCare V3");
    oled.setCursor(0,20);
    oled.println(" IV Drop Detector");
    oled.setCursor(0,30);
    oled.println(" WiFi Version");
    oled.setCursor(0,40);
    oled.println("--------------------");
    oled.display();
    delay(2000);

    // Connect to WiFi
    oled.clearDisplay();
    oled.setTextSize(1);
    oled.setCursor(0,10);
    oled.println("Connecting WiFi...");
    oled.display();
    connectWiFi();

    if (WiFi.status() == WL_CONNECTED) {
        deviceIP = WiFi.localIP().toString();
        oled.setCursor(0,20);
        oled.println("Connected!");
        oled.setCursor(0,30);
        oled.println(deviceIP.c_str());
        oled.display();
        delay(2000);
    }
    else {
      oled.setCursor(0,20);
      oled.println("WiFi Failed");
      oled.setCursor(0,30);
      oled.println("Offline Mode");
      oled.display();
      delay(2000);
    }

    // Setup web server
    setupWebServer();

    // --- Drop Sensor Interrupt ---
    pinMode(DROP_PIN, INPUT_PULLUP); // active LOW
    attachInterrupt(digitalPinToInterrupt(DROP_PIN),
           onDropDetected, FALLING);

    // --- Get medicine volume and duration from user ---
    medicineVolume = getUserInput("Med Vol (mL):");
    durationMinutes = getUserInput("Duration (min):");

    // Calculate preset drop rate: drops per minute
    presetDropRate = (medicineVolume * 20.0) / durationMinutes;

    oled.clearDisplay();
    oled.setTextSize(2);
    oled.setCursor(0,0);
    oled.println("Preset DPM");
    oled.setTextSize(3);
    oled.setCursor(0,30);
    oled.println(presetDropRate);
    oled.display();

    delay(3000); // show preset rate for 3 seconds

    starttime = millis(); // Start timing

    Serial.println("\n========================================");
    Serial.println("Device Ready!");
    Serial.println("Device Number: " + deviceNumber);
    Serial.println("IP Address: " + deviceIP);
    Serial.println("Expected Drop Rate: " + String(presetDropRate,
         1) + " drops/min");
    Serial.println("Data Endpoint: http://" + deviceIP + "/data");
    Serial.println("========================================\n");
}

void loop() {
    // --- Handle web server requests ---
    server.handleClient();

    // --- Keypad handling ---
    char key = keypad.getKey();
    if (key == '*') {
      oled.clearDisplay();
      oled.setTextSize(2);
      oled.setCursor(0,0);
      oled.println("Rebooting...");
      oled.display();
      digitalWrite(BUZZER_PIN, LOW);
      delay(500);
      esp_restart();
    }
    if (key == '#') {
    dropCount = 0; // reset counter
    display.clear();
    oled.clearDisplay();
    oled.setTextSize(2);
    oled.setCursor(0,0);
    oled.println("Reset Done");
    digitalWrite(BUZZER_PIN, LOW);
    oled.display();
    delay(100);
}

    unsigned long currentMillis = millis();

    // --- Calculate instantaneous current DPM ---
    currentDPM = calculateInstantDPM();
    float percentageError = 0;

    if(presetDropRate > 0){
        percentageError = fabs(currentDPM - presetDropRate) /
               presetDropRate * 100.0;
    }
        // Track when deviation starts
    static bool deviationStarted = false;

    if (percentageError >= 10) {
        if (!deviationStarted) {
            overLimitStart = millis(); // mark start of deviation
            deviationStarted = true;
        }
    } else {
        deviationStarted = false; // back to normal
    }

    if(millis() < alarmBlockUntil){
          digitalWrite(BUZZER_PIN, LOW);
          oled.clearDisplay();
          oled.setTextSize(2);
          oled.setCursor(0,44);
          oled.println("ADJUSTING");
          oled.display();

        return;
      }

    // --- Update displays every 0.5 second ---
    if (currentMillis - previousDisplayMillis >= displayInterval)
           {
        // TM1637: current DPM
        display.showNumberDec((int)currentDPM, false);

        // OLED: preset DPM, current DPM, and WiFi status
        oled.clearDisplay();
        oled.setTextSize(1);
        oled.setCursor(0,0);
        oled.println("Preset DPM:");
        oled.setTextSize(3);
        oled.setCursor(0,18);
        oled.println((int)presetDropRate);
        oled.setTextSize(1);
        oled.setCursor(0,40);
        oled.println("Current DPM:");
        oled.setTextSize(2);
        oled.setCursor(0,50);
        oled.println((int)currentDPM);

        // WiFi status at bottom
        oled.setTextSize(1);
        oled.setCursor(0,54);
        if(WiFi.status() == WL_CONNECTED) {
            // oled.println("WiFi: " + deviceIP.substring(deviceIP.
                lastIndexOf('.') + 1));
        } else {
            oled.println("WiFi: Off");
        }
        oled.display();

        previousDisplayMillis = currentMillis;

    }

    // not enough drops  do nothing
    if(!enoughDrops){
        digitalWrite(BUZZER_PIN, LOW);
        return;
    }
```

```
    if (percentageError < 10) {
        // NORMAL
        digitalWrite(BUZZER_PIN, LOW);
    }
    else if (percentageError < 20) {
        // WARNING
        if (millis() - melodyPrev > 500) {
            melodyPrev = millis();
            digitalWrite(BUZZER_PIN, !digitalRead(BUZZER_PIN));
        }
    }
    else {
        // CRITICAL
        unsigned long devTime = millis() - overLimitStart;

        if (devTime >= deviationDelay + patternDuration) {
            playPanicPattern(); // panic
        }
        else if (devTime >= deviationDelay) {
            playMelodyPattern(); // melody stage
        }
        else {
            digitalWrite(BUZZER_PIN, LOW); // initial deviation
        }
    }

}

// ========== WiFi CONNECTION ==========
void connectWiFi() {
  Serial.print("Connecting to WiFi: ");
  Serial.println(ssid);

  // Ensure station mode before connecting
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  int attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 30) {
    delay(500);
    Serial.print(".");
    attempts++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\n WiFi Connected!");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\n WiFi Connection Failed!");
    Serial.println("Device will work in offline mode.");
  }
}

// ========== WEB SERVER SETUP ==========
void setupWebServer() {
  server.on("/", handleRoot);
  server.on("/data", HTTP_GET, handleGetData);
  server.onNotFound(handleNotFound);
  server.begin();
  Serial.println(" Web server started on port 80");
}

// ========== WEB SERVER HANDLERS ==========
void handleRoot() {
  String html = "<!DOCTYPE html><html><head><title>ESP32 V3 IV Monitor</title>";
  html += "<meta name='viewport' content='width=device-width, initial-scale=1'>";
  html += "<style>body{font-family:Arial;padding:20px;background:#f0f0f0;margin:0;}";
  html += ".container{background:white;padding:20px;border-radius:8px;max-width:600px;margin:0 auto;box-shadow:0 2px 4px rgba(0,0,0,0.1);}";
  html += "h1{color:#203f76;margin-top:0;}";
  html += "table{width:100%;border-collapse:collapse;margin:20px 0;}";
  html += "td{padding:12px 8px;border-bottom:1px solid #ddd;}";
  html += "td:first-child{font-weight:bold;color:#203f76;width:40%;}";
  html += ".status{display:inline-block;padding:4px 12px;border-radius:12px;font-size:14px;}";
  html += ".online{background:#d4edda;color:#155724;}";
  html += ".offline{background:#f8d7da;color:#721c24;}";
  html += ".warning{background:#fff3cd;color:#856404;}";
  html += "a{color:#57bdd4;text-decoration:none;font-weight:bold;}";
  html += "a:hover{text-decoration:underline;}</style></head>";
  html += "<body><div class='container'>";
  html += "<h1> InfuCare IV Monitor V3</h1>";
  html += "<table>";
  html += "<tr><td>Device Number:</td><td>" + deviceNumber + "</td></tr>";
  html += "<tr><td>IP Address:</td><td>" + deviceIP + "</td></tr>";
  html += "<tr><td>Status:</td><td><span class='status online'>Active</span></td></tr>";
  html += "<tr><td>WiFi:</td><td><span class='status " + String(WiFi.status() == WL_CONNECTED ? "online'> Connected" : "offline'> Disconnected") + "</span></td></tr>";
  html += "<tr><td>Expected Rate:</td><td><strong>" + String(presetDropRate, 1) + "</strong> drops/min</td></tr>";
  html += "<tr><td>Current Rate:</td><td><strong>" + String(currentDPM, 1) + "</strong> drops/min</td></tr>";
  html += "<tr><td>Total Drops:</td><td><strong>" + String(dropCount) + "</strong></td></tr>";
  html += "<tr><td>Volume Setting:</td><td>" + String(medicineVolume) + " ml</td></tr>";
  html += "<tr><td>Time Setting:</td><td>" + String(durationMinutes) + " minutes</td></tr>";

  float elapsedSeconds = (millis() - starttime) / 1000.0;
  int minutes = int(elapsedSeconds / 60);
  int seconds = int(elapsedSeconds) % 60;
  html += "<tr><td>Running Time:</td><td>" + String(minutes) + "m " + String(seconds) + "s</td></tr>";

  float deviation = abs(currentDPM - presetDropRate);
  float deviationPercent = (presetDropRate > 0) ? (deviation / presetDropRate) * 100 : 0;
  html += "<tr><td>Deviation:</td><td>" + String(deviation, 1) + " drops/min (" + String(deviationPercent, 1) + "%)</td></tr>";

  // Alarm status
  String alarmStatus = "Normal";
  if(currentDPM > presetDropRate) {
    unsigned long devTime = millis() - overLimitStart;
    if(devTime >= deviationDelay + patternDuration) {
      alarmStatus = "PANIC ALARM";
    } else if(devTime >= deviationDelay) {
      alarmStatus = "Melody Alert";
    } else {
      alarmStatus = "Deviation Detected";
    }
  }
  html += "<tr><td>Alarm Status:</td><td><span class='status " + String(alarmStatus == "Normal" ? "online'>" : "warning'>") + alarmStatus + "</span></td></tr>";

  html += "</table>";
  html += "<p style='margin-top:20px;text-align:center;'><a href='/data'> View JSON Data</a></p>";
  html += "<p style='text-align:center;color:#666;font-size:12px;margin-top:30px;'>InfuCare Team  2025 - V3</p>";
  html += "</div></body></html>";

  server.send(200, "text/html", html);
  Serial.println(" Web page served");
}

void handleGetData() {
  // Prepare JSON response
  StaticJsonDocument<512> doc;
  doc["device_number"] = deviceNumber;
  doc["drop_rate"] = currentDPM;
  doc["drop_count"] = dropCount;
  doc["expected_drop_rate"] = presetDropRate;
  doc["timestamp"] = String(millis());
  doc["device_status"] = "online";
  doc["wifi_status"] = (WiFi.status() == WL_CONNECTED) ? "connected" : "disconnected";
  doc["ip_address"] = deviceIP;

  // Calculate deviation
  float deviation = abs(currentDPM - presetDropRate);
  float deviationPercent = (presetDropRate > 0) ? (deviation / presetDropRate) * 100 : 0;
  doc["deviation"] = deviation;
  doc["deviation_percent"] = deviationPercent;

  // Add settings
  doc["volume_setting"] = medicineVolume;
  doc["time_setting"] = durationMinutes;

  // Add runtime
  float elapsedSeconds = (millis() - starttime) / 1000.0;
  doc["runtime_seconds"] = elapsedSeconds;

  // Alarm status
  String alarmStatus = "normal";
  if(currentDPM > presetDropRate) {
    unsigned long devTime = millis() - overLimitStart;
    if(devTime >= deviationDelay + patternDuration) {
      alarmStatus = "panic";
    } else if(devTime >= deviationDelay) {
      alarmStatus = "melody";
    } else {
      alarmStatus = "deviation";
    }
  }
  doc["alarm_status"] = alarmStatus;

  String jsonData;
  serializeJson(doc, jsonData);

  // Send CORS headers
  server.sendHeader("Access-Control-Allow-Origin", "*");
```

```
  server.sendHeader("Access-Control-Allow-Methods", "GET");
  server.send(200, "application/json", jsonData);

  Serial.println(" Data request served - Rate: " + String(
        currentDPM, 1) + " d/m");
}

void handleNotFound() {
  server.send(404, "text/plain", "404: Not Found");
}
```

# References

[1] P. G. Yock, S. Zenios, J. Makower, T. J. Brinton, U. N. Kumar, F. T. J. Watkins, L. Denend, T. M. Krummel, and C. Q. Kurihara, Eds., Biodesign: The Process of Innovating Medical Technologies, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2015

[2] O. Hoffman and O. Bacon, "Infusion Pumps," in Making Healthcare Safer III: A Critical Analysis of Existing and Emerging Patient Safety Practices, K. K. Hall, S. Shoemaker-Hunt, L. Hoffman, et al., Eds. Rockville, MD, USA: Agency for Healthcare Research and Quality (US), Mar. 2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK555506/

[3] B. Jung, K. S. Seo, S. J. Kwon, K. Lee, S. Hong, H. Seo, G. Y. Kim, G. M. Park, J. Jeong, and S. Seo, "Efficacy evaluation of syringe pump developed for continuous drug infusion," J. Dent. Anesth. Pain Med., vol. 16, no. 4, pp. 303–307, Dec. 2016. doi: 10.17245/jdapm.2016.16.4.303. PMID: 28879319; PMCID: PMC5564196. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC5564196/