International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 04 | Apr 2024 www.irjet.net p-ISSN: 2395-0072

Design and Development of Cloud Connected Infusion Monitor

C R Nikethan[1], Ashwini V[2]

[1]P G Student
[2]Professor, Dept. of Electronics and Communication Engineering, BMS College of Engineering, Karnataka, India

---

Abstract - The development of a cloud-connected infusion monitor and its design are described in this study. Infusion therapy plays a critical role in modern healthcare, requiring precise monitoring to ensure patient safety and optimal treatment outcomes. Traditional infusion monitors lack seamless connectivity and real-time data sharing, limiting their effectiveness in remote patient management and data analysis. To address this gap, we propose a novel infusion monitor that integrates sensors, data processing capabilities, and cloud connectivity. The monitor continuously measures infusion parameters such as flow rate, volume administered, and transmitting this data securely to a cloud-based platform. This enables healthcare providers to remotely monitor patients' infusion sessions, receive real-time alerts, and access historical data for informed decision-making. The development process involves hardware design, sensor integration, data processing algorithms, and secure cloud communication protocols. The resulting cloud-connected infusion monitor offers enhanced patient safety, healthcare workflow optimization, and opportunities for data-driven insights into infusion therapy management. This innovation has the potential to revolutionize infusion therapy by fostering greater collaboration between healthcare providers and improving patient care through intelligent and connected monitoring systems.

Key Words: Infusion Solution, Intravenous, Wireless Module, ESP8266, Visual Studio.

1.INTRODUCTION

In modern healthcare settings, the accurate administration of intravenous (IV) fluids and medications is critical for patient well-being. Infusion monitors play an important role in maintaining the precision and safety of infusion systems by providing real-time monitoring and feedback on various parameters. This technical introduction explores the key components and functionalities of infusion monitors, highlighting their importance in delivering optimal patient care. The fig 1 shows the components involved in a standard gravitational drip. As showcased. As you can see above, it consists of container, drip chamber, and the roller clamp which is utilized to control the flow of the fluid induced.

Infusion Monitoring System: An infusion monitoring system comprises a combination of hardware and software components designed to ensure precise control and monitoring of IV fluid and medication administration. It typically includes a monitor or display unit, sensors, connectivity modules, and software interfaces.

Monitoring Parameters: Infusion monitors are equipped to measure and monitor several essential parameters, including flow rate, volume infused, pressure, air detection, and occlusion detection. These parameters enable healthcare professionals to closely monitor the progress and effectiveness of the infusion process, ensuring that the prescribed dosage is accurately delivered to the patient.

Flow Rate Monitoring: Flow rate monitoring is among the primary functions of an infusion monitor. It measures the rate at which the IV fluid or medication is given to the patient. By continuously monitoring the flow rate, the infusion monitor can detect any variations or deviations from the prescribed rate, allowing healthcare providers to take immediate corrective actions.

Volume Infused Calculation: Accurate calculation of the volume of fluid or medication infused is crucial in managing patient treatment plans. Infusion monitors employ advanced algorithms and sensors to precisely calculate the volume of fluid delivered over a specified time period. This information helps healthcare professionals track the progress of the infusion, adjust dosages if necessary, and ensure adherence to treatment protocols.

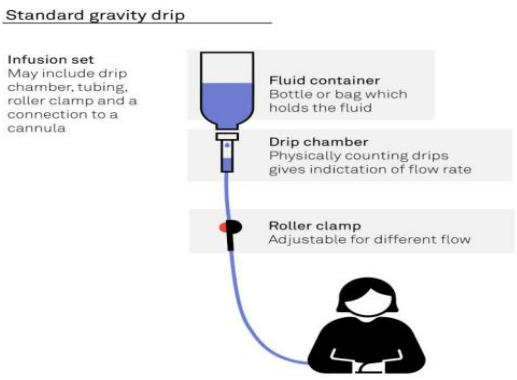

Fig 1. Standard Gravity Drip

Pressure Monitoring: Pressure monitoring is essential to prevent potential complications during the infusion process. Infusion monitors are equipped using pressure sensors that detect and alert healthcare providers to abnormal pressure levels, such as occlusions or blockages in the IV line. By monitoring pressure, infusion monitors enhance patient safety by minimizing the risk of infiltration or extravasation.

Air and Occlusion Detection: Air detection and occlusion detection capabilities are critical features of infusion monitors. Air detection sensors enable the system to identify the presence of air bubbles in the IV line, triggering alarms and preventing potentially harmful air embolisms. Occlusion detection sensors monitor the pressure exerted on the IV line and alert healthcare providers to potential blockages orkinks, allowing for prompt intervention.

Alarms and Notifications: Infusion monitors provide real-time alarms and notifications to healthcare professionals in the event of deviations from established parameters. These alerts prompt immediate attention, preventing errors, and facilitating timely intervention. Additionally, infusion monitors may offer connectivity options to send alerts to centralized monitoring systems or sync with the electronic health record (EHR) systems, facilitating comprehensive patient care.

Infusion monitors represent a crucial component in medical infusion systems, enabling healthcare providers to ensure precision and safety during the administration of IV fluids and medications. By monitoring parameters such as flow rate, volume infused, pressure, and detecting air or occlusions, these advanced monitoring systems enhance patient care and contribute to improved patient outcomes. The integration of infusion monitors with other healthcare technologies further streamlines care delivery and promotes efficient treatment practices.

The research says that the error happening due to infusion leads to death and it is placed in 3rd place for leading cause of death behind heart diseases and cancer. Around 4,40,000 deaths happen per year due to medical error. Around 60% of IV infusion contained one or more errors. So, to overcome all these errors the proposed system will be more beneficial.

1.1 Calculation Flow rate:

Drip factor is given when what kind of IV set we use. There are two IV sets Micro and Macro. Micro is 60 gtt/ml and Macro is 10, 15, 20 gtt/ml.

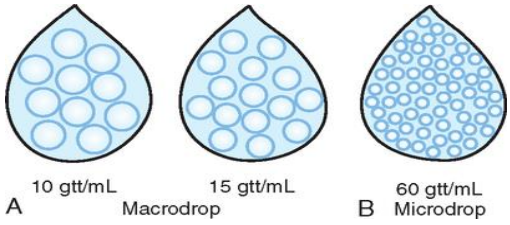

Fig 2. Drop Factor

$$\frac{\text{Amount of fluid (mL)}}{\text{Total time of infusion (min)}} \text{ x Drop factor (gtts/mL) = IV infusion rate (gtts/min)}$$

2. PROPOSED METHODOLOGY

The prototype of my product is shown in fig 3 the processor and electronics component is placed and fixed to the stand to make sure it is protected from theft. The detection patch is connected up to the drip chamber and it displays the details. The detection and processor is connected by the wire. Once the sensor detects the change it sends data to processor. The processor makes sure the required data is displayed in the device display and it is being transferred to cloud. The reason for transferring data to the cloud is to make sure to take responsibility for the errors (MedicoLegal purpose).

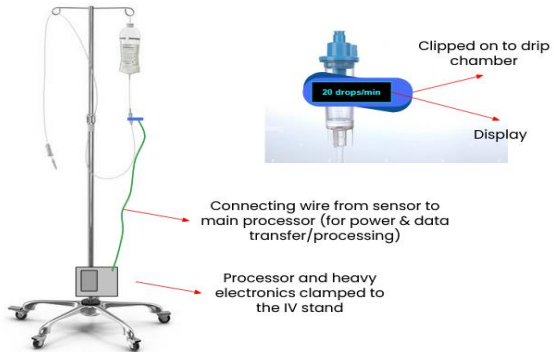

Fig 3. Proposed Methodology

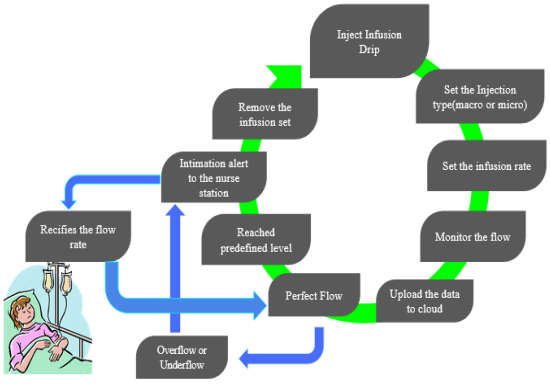

Fig 4. Flow chart of the steps involved in the project

2.1 Socket Programming

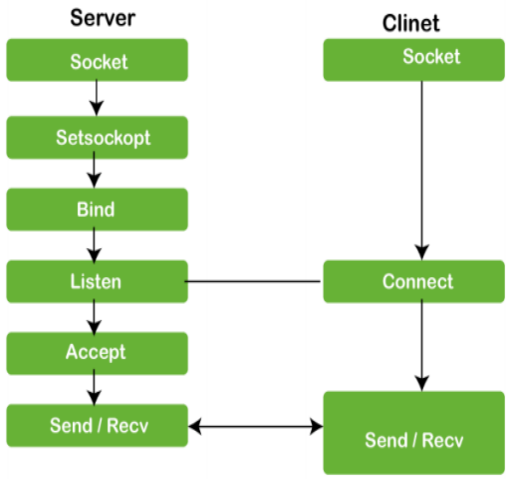

Fig 5. Socket Programming Connection Between Server and Client

Socket Creation:

The process begins with the server creating a socket. A socket is a communication endpoint that allows data exchange between devices over a network.

Setsockopt:

After creating the socket, the server sets socket options. These options configure various properties of the socket, such as its behavior and communication parameters.

Bind:

The server binds the socket to a specific address and port. This step associates the socket with a network interface and makes it ready to listen for incoming connections.

Listening:

Once bound, the server enters a listening state. It actively waits for incoming connection requests from clients.

Accepting Connections:

When a client initiates a connection, the server accepts the request. This establishes a connection between the server and the client.

Data Exchange:

With the connection established, both server and client can send and receive data through their respective sockets using the "Send" and "Receive" operations.

Client Side:

Socket Creation:

On the client side, a similar process occurs. The client creates its own socket.

Connecting to Server:

The client initiates a connection by specifying the server's address and port. It sends a connection request.

Data Exchange:

Once connected, the client can also send and receive data through its socket.

2.2 Google Cloud Run

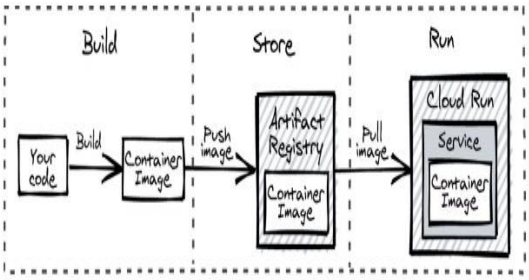

Fig 6. Cloud Run Process

Build:

You start with your source code (depicted as "Your code"). This code is then built into a container image. The arrow labeled "Push Image" indicates that the built container image needs to be pushed to a storage location.

Store:

The "Artifact Registry Container Image" box represents where the pushed container images are stored. When needed, the stored image can be retrieved from this registry.

Run:

The "Cloud Run Service Container Image" box indicates where and how the pulled container images are executed or run.

3. PROGRESS

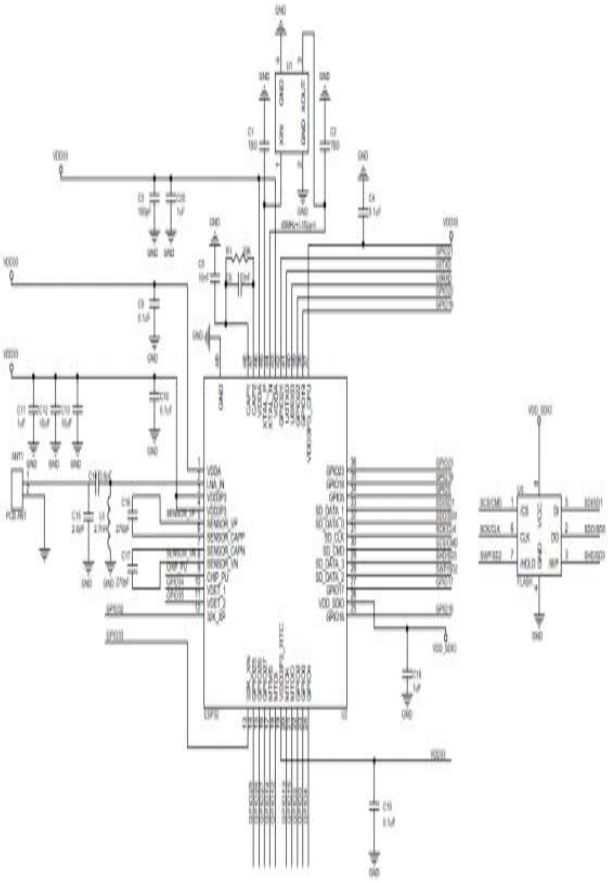

Fig 7. Circuit Diagram of ESP Wroom 32 Module

In the 1st stage implementing the connection between the microcontroller and the sensor. In this stage interrupt is used to implement the connection. Once the code was uploaded there was lots of bounce happening. Timer interrupt was implemented to induce debounce. Implementation of RTC to cross verify the data received with the drop rate. Assigning GMT value and set RTC time manually. To get the results getTime is used to receive the data at the output. Implementing client server communication. To understand the client-server communication, implement socket programming. Client is the microcontroller and server are another pc which is connected through the same LAN. Next step is to transfer data from client (microcontroller) to cloud through http protocol.

4. RESULTS

4.1 Output of sensor

```
Count:      9     Count_timer:    5529    12:40:55,
Count:      10    Count_timer:    6008    12:41:00,
Count:      11    Count_timer:    6337    12:41:03,
Count:      12    Count_timer:    6502    12:41:05,
Count:      13    Count_timer:    6691    12:41:06,
Count:      14    Count_timer:    6879    12:41:08,
Count:      15    Count_timer:    7208    12:41:12,
Count:      16    Count_timer:    7331    12:41:13,
Count:      17    Count_timer:    7454    12:41:14,
Count:      18    Count_timer:    7468    12:41:14,
Count:      19    Count_timer:    7612    12:41:16,
Count:      20    Count_timer:    7760    12:41:17,
Count:      21    Count_timer:    8045    12:41:20,
Count:      22    Count_timer:    8146    12:41:21,
```

Fig 8. Output of Sensor in Serial Monitor

The image presented above captures a moment when a successful socket programming connection was established. In this context, the Arduino serial monitor comes into play as it reveals the outcome of sensor data retrieval. Specifically, the displayed information includes the count, internal timer interrupt status, and RealTime Clock (RTC) data. This data, as per the implemented code, reflects the sensor's functioning and the synchronization with various system.

4.2 Server-Side Output

```
========== RESTART: C:/Users/carditek/Desktop/v0.2_server.py ==========
Waiting for a connection...
connected
Drop Count
b'1'
Drop Count
b'2'
Drop Count
b'3'
Drop Count
b'4'
Drop Count
b'5'
Drop Count
b'6'
Drop Count
b'7'
Drop Count
b'8'
Drop Count
b'9'
Drop Count
b'10'
Drop Count
b'11'
Drop Count
b'12'
Drop Count
b'13'
Drop Count
b'14'
Drop Count
b'15'
Drop Count
b'16'
Drop Count
b'17'
Drop Count
b'18'
Drop Count
```

Fig 9. Output from the Server Side

Upon the successful establishment of socket programming, the server side enters a waiting state, poised to facilitate a connection between the client and itself through the utilization of port number 80. Port 80, a widely recognized standard, is predominantly employed for the transmission of data via the Hypertext Transfer Protocol (HTTP). The initiation of this connection is primarily driven by the use of the client's IP address, allowing for seamless communication between the two endpoints. This crucial interaction forms the backbone of web-based data exchange, enabling clients to request and receive web content hosted on the server. The utilization of port 80 and IP addressing underscores the fundamental principles of network communication, enabling the flow of information across the internet and the World Wide Web.

4.3 Cloud Output

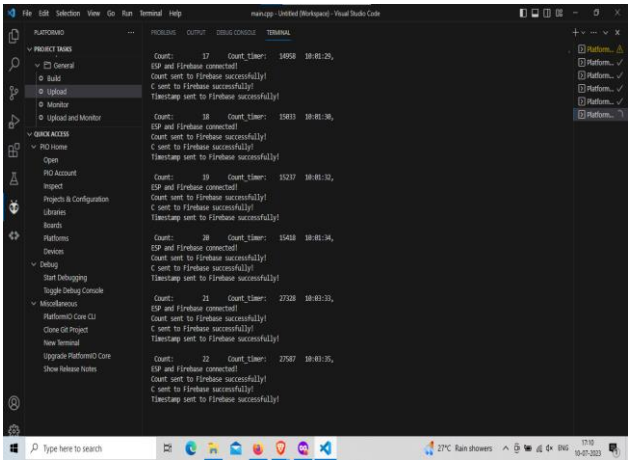

Fig 10. After Cloud Connection Serial Monitor Output

Following the successful establishment of the cloud connection, facilitated by the utilization of API keys and specifying the designated path for data transfer, the Visual Studio Code (VSCode) serial terminal provides a real-time display of sensor output data enriched with the essential comments incorporated in the codebase. This interactive terminal interface enhances the monitoring and debugging process for developers.

With the microcontroller seamlessly connected to the cloud, the actual data transfer process commences. Image 11 vividly illustrates how this data is elegantly showcased within Firebase, a popular cloud database platform. Firebase serves as the conduit for storing, managing, and visualizing the transmitted data, rendering it accessible and organized for various analytics, applications, and downstream processes. This image captures the pivotal moment when the data integration between the microcontroller and the cloud takes shape, bridging the physical and digital realms and enabling a wide array of possibilities in the realm of IoT and data-driven applications.

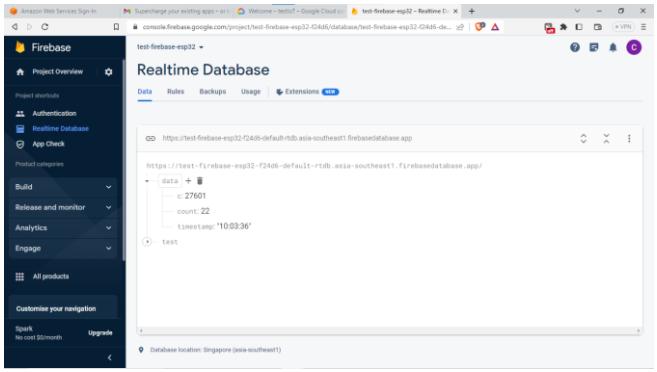

Fig 11. After Cloud Connection Cloud Firebase Output

5. CONCLUSIONS

The Cloud Connected Infusion Monitor developed in this project offers significant benefits for healthcare delivery. By integrating infusion monitoring with cloud connectivity, we've improved patient safety through real-time data transmission and remote monitoring. This system enhances efficiency, scalability, and data management in healthcare settings. Moving forward, further advancements can expand its functionality and integration with existing systems to maximize its impact on patient care.

6. REFERENCES

[1] K. R. Rani, N. Shabana, P. Tanmayee, S. Loganathan, and G. Velmathi, "Smart drip infusion monitoring system for instant alert through nRF24L01," *ICNETS2*, 2017.

[2] A. Cataldo, G. Cannazza, N. Giaquinto, A. Trotta, and G. Andria, "Development of a remote system for real-time control of intravenous drip infusions," *IEEE*, 2011.

[3] German, J.D., Mina, J.K.P., Alfonso, C.M.N., Yang, K.-H., "A study on shortage of hospital beds in the Philippines using system dynamics," *ICIEA*, 2018.

[4]T.L. Rodziewicz, and J.E. Hipskind, Medical Error Prevention. Treasure Island(FL): StarPearls Publishing, 2018.

[5] Design of Wireless Infusion Monitor Based on Bluetooth 4.0, Shilin Wanf School of Mechanical, Electrical and Information Engineering Shandong University Weihai, China, Baochen Jiang* School of Mechanical, Electrical and Information Engineering Shandong University Weihai, China, *IEEE*, 2018.