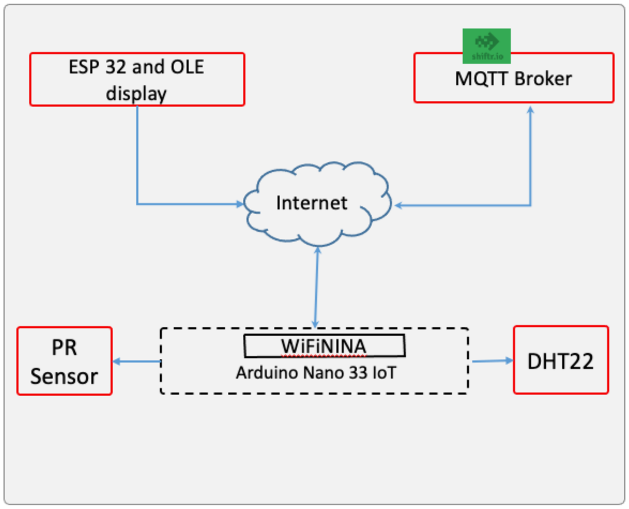
Smart Baby monitor Project report

Shivakumar Nyamagoud
snyamago@gmu.edu
G01405418

Abstract:

This project aims to a comprehensive monitoring system for babies, employing a microcontroller device to oversee their well-being. Utilizing components such as the Arduino Nano IoT 33, ESP32, DHT sensor, motion sensor (PIR), and OLED display, the system integrates functionalities to monitor motion, room temperature, and humidity. Leveraging WiFi and MQTT protocols, the microcontroller orchestrates concurrent threads to execute various tasks seamlessly.

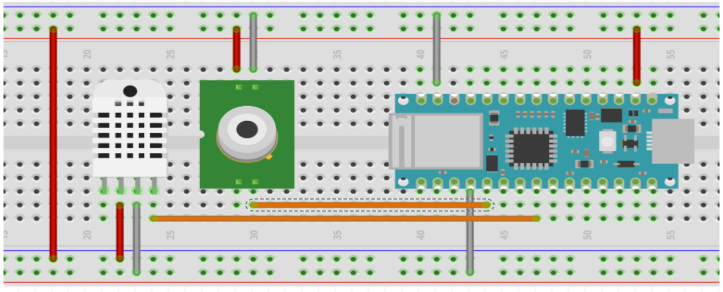
One thread periodically collects temperature and humidity data, transmitting it via MQTT to a remote device. Concurrently, another thread monitors baby motion and sends notifications upon detection. These threads interface with an ESP32 device featuring an OLED display, presenting real-time information. Additionally, the system uploads data to the web for remote access via smartphones and other microcontroller devices. With its multi-threaded design and MQTT communication, the system ensures efficient monitoring and timely alerts, enhancing baby care and parental peace of mind.



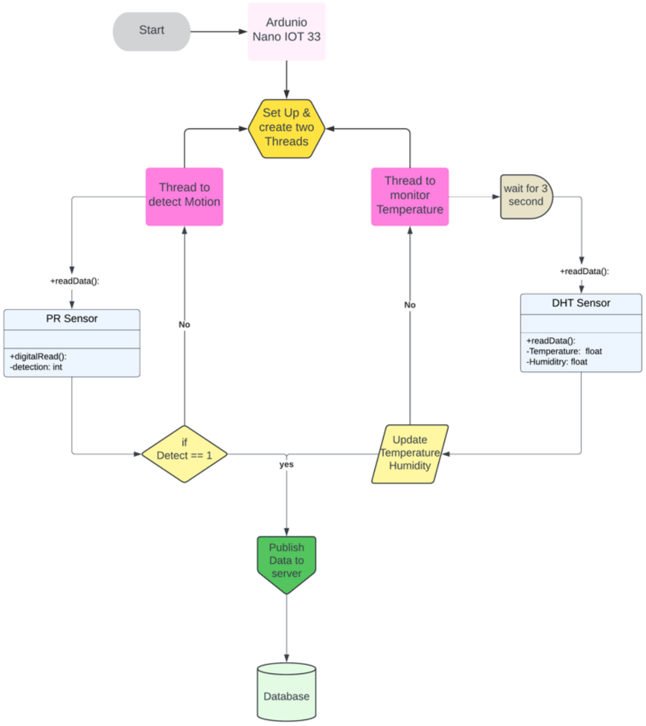
The above Block diagram I divide this project in 3 parts to show the flow of design.

1) Arduino nano IOT 33,
2) ESP 32,
3) Communication Protocol.

1) Arduino nano IOT 33

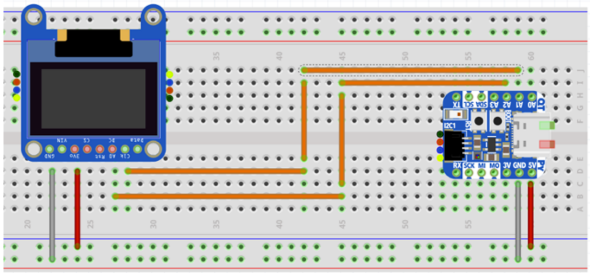


The Arduino will call setup function where it assigns pins to DHT sensor and PIR sensor and connects to WiFi, creates topic to publish for Mqtt. Finally creates one schedular thread/task for independent data reading and this thread will run concurrently to perform its tasks. The schedular thread which reads temperature and humidity data for every three-second intervals (static time) and published the data to Mqtt server this acts as time trigger task. And second thread/task the Arduino loop function. which reads detect flag which is set by Arduino interrupt and if the motion detects then that thread publishes the data to Mqtt server data base.

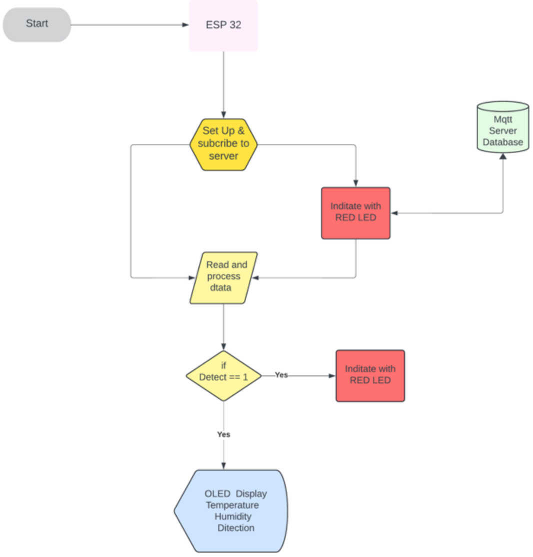


This is the flowchart for the Arduino Nano Iot 33 module.

2) ESP 32



ESP32 device, connects to WiFi, then subscribes to the Mqtt servers topics. Whenever the data publishes from Arduino, that will display in ESP32 0.96" OLED display which communicates through I2C to display collected information. If PIR detects motion, then blink a light if the baby wakes up until further user input. Moreover, this data will be uploaded onto the web, enabling access from smartphones and other microcontroller devices equipped with displays.



This is the flowchart for the ESP 32 module.

3) Protocol:

1. **MQTT:** it is a messaging protocol. It is used to design for transferring messages and uses a publish and subscribe model where a publisher publishes messages on a topic and a subscriber must subscribe to that topic to view the message. In the MQTT it possible to send messages to 1 or multiple clients and at the same time we can subscribe the multiple message topics from the MQTT clients. All clients can publish (broadcast) and subscribe (receive).

2. **I2C:** Inter-Integrated Circuit, is a synchronous serial communication protocol used to connect multiple peripheral devices. It uses two wires: a serial data line (SDA) and a serial clock line (SCL). For communication. In this project I2C is used to communicate between the ESP32 and OLED display.

What is Real-time Embedded System here?

1. Considering both Hardware ESP32 and Arduino are microcontrollers which responds to external events or sensors and perform tasks within predictable time constraints. The devices support Interrupt handling, scheduling, Timer and Task Management and Low Latency Communication.
2. As a task scheduling in this project, I have used the schedule library to schedule the Temperature and humidity reading thread every three second and publish the data to Mqtt.
3. For the PIR sensor i used the Interrupt APIs to configure the interrupts in Arduino. In project setting up an interrupt on a specified digital pin for a PIR motion sensor to execute a specific function when a particular event occurs on the specified pin. Means when the state of the pin changes, the detect function will be called to handle the motion detection.
4. Time driven and Event driven task handling, the Reading temperature and humidity is time driven. And detecting the baby motion by PIR sensor is event driven task.
5. Using the Realtime communication Protocols like I2C and Mqtt.

Implementation:

In setup function using scheduler called the Read Temperature function to read DHT sensor data for every three second

```
Scheduler.startLoop(ReadTemperature, 10000);
```

Configured the PIR sensor pin to Arduino interrupt handler to execute a specific function when a particular event occurs on the specified pin

```
pinMode(PIR_Interrupt_pin, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(PIR_Interrupt_pin), detectPIR, CHANGE);
```

To mutual execution of Arduino loop and schedular thread, used yield() in both task so that both will gets it execution in the Arduino board.

```
Scheduler.yield();
```

Reading the Temperature and Humidity using dht22 library.

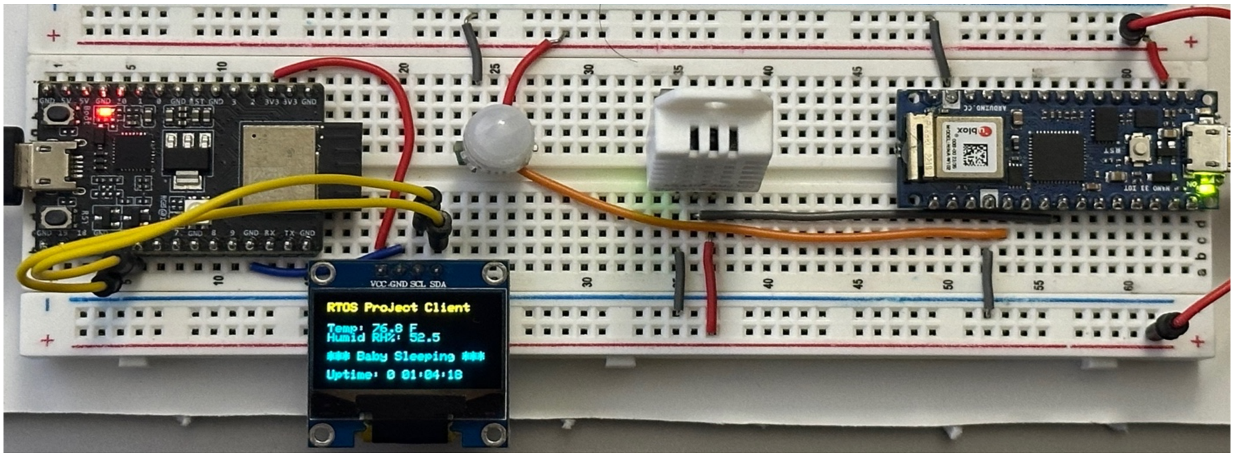
```
errDHT22 = dht22.read2(&Temp_data, &Humid_data, NULL);
```

If any data changes in temperature, humidity and Baby Motion publishing this on Mqtt protocol and receiving it in ESP32 device.

```
babyAwake = true;
sprintf(PIRBuffer, "*** Baby wakeup ***");
mqttdataPublish = String(PIRBuffer);
mqttClient.publish(topicPub3, mqttdataPublish);
```

Some other implementation also done in ESP32 design and implementation, for reference attaching the code in submission.

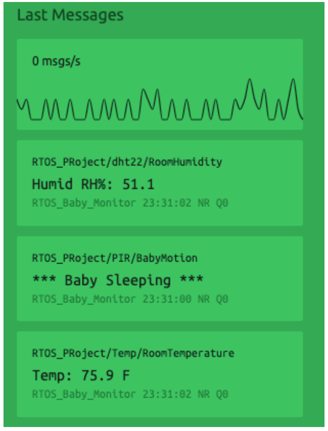
Testing: Performed most of the testing on both devices working as expected. Most of the testing results demonstrated in the video explanation.



RTOS Project Client
Temp: 76.8 F
Humid RH%: 52.5
Baby Sleeping
Uptime: 0 01:04:18

In the above image shows the actual circuit connection and ROTS system up and running.
Note: I have only one breadboard so both circuit connection made on same bread board with different USB power supply.

Mqtt visualization



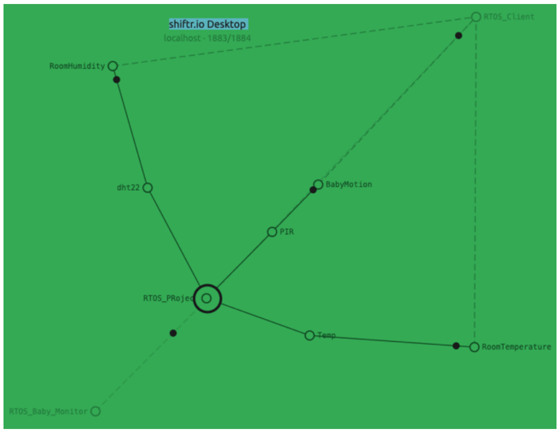
Last Messages

0 msgs/s

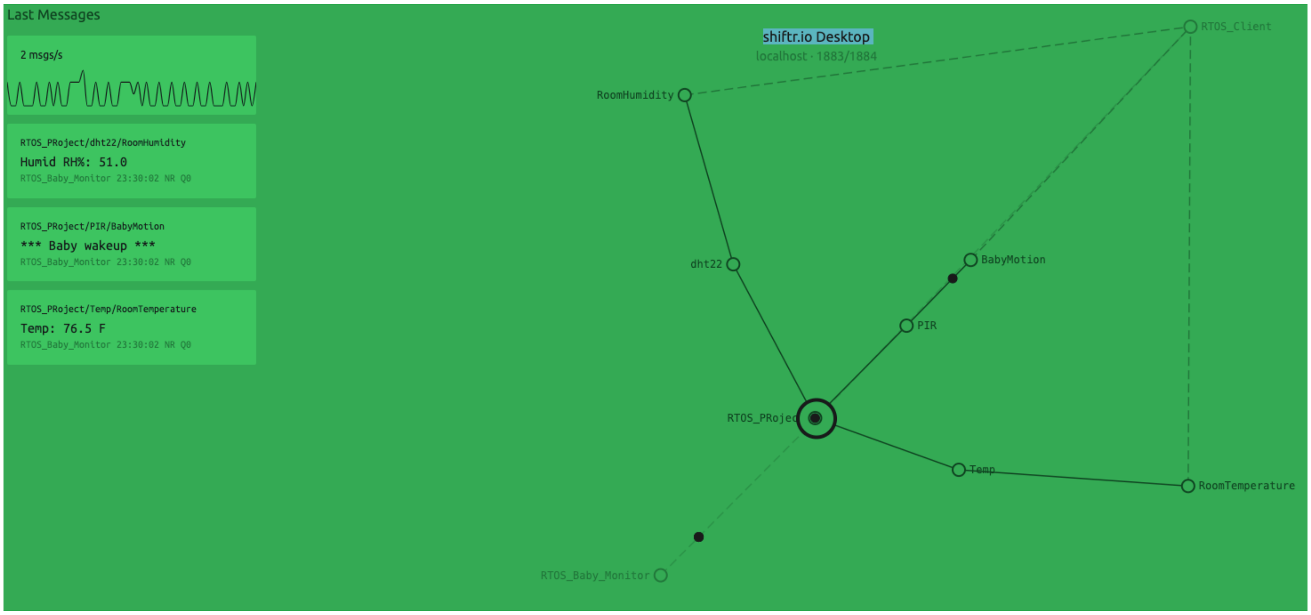
RTOS_PRoject/dht22/RoomHumidity
Humid RH%: 51.1
RTOS_Baby_Monitor 23:31:02 NR Q0

RTOS_PRoject/PIR/BabyMotion
*** Baby Sleeping ***
RTOS_Baby_Monitor 23:31:00 NR Q0

RTOS_PRoject/Temp/RoomTemperature
Temp: 75.9 F
RTOS_Baby_Monitor 23:31:02 NR Q0



Mqtt Shifter IO desktop view, here we can see both ESP32 and Arduino Device publish and subscribed data being visualized. Left images shows the data being published and right image shows the communication path.



Last Messages

2 msgs/s

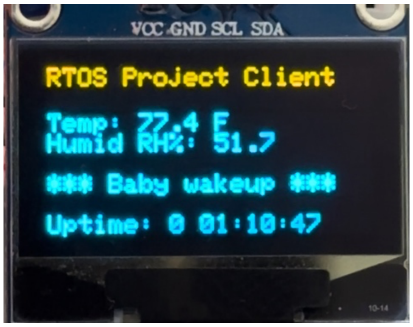
RTOS_PRoject/dht22/RoomHumidity
Humid RH%: 51.0
RTOS_Baby_Monitor 23:30:02 NR Q0

RTOS_PRoject/PIR/BabyMotion
*** Baby wakeup ***
RTOS_Baby_Monitor 23:30:02 NR Q0

RTOS_PRoject/Temp/RoomTemperature
Temp: 76.5 F
RTOS_Baby_Monitor 23:30:02 NR Q0

Test cases and results: **In the video has more than all test cases**

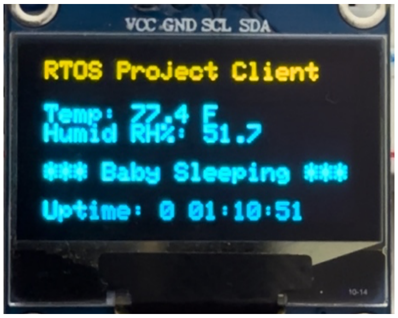
1. Tested with making continues motion, so that systems detected each movement and published the message saying Baby wakeup.



```
RTOS Project Client
Temp: 77.4 F
Humid RH%: 51.7
*** Baby wakeup ***
Uptime: 0 01:10:47
```

When PIR detects motion

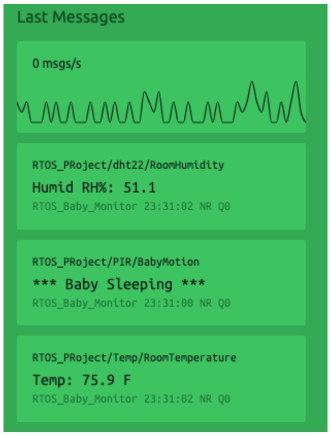
2. Tested with no motion, so that systems didn't detect any movement and published the message saying Baby sleeping only once.



```
RTOS Project Client
Temp: 77.4 F
Humid RH%: 51.7
*** Baby Sleeping ***
Uptime: 0 01:10:51
```

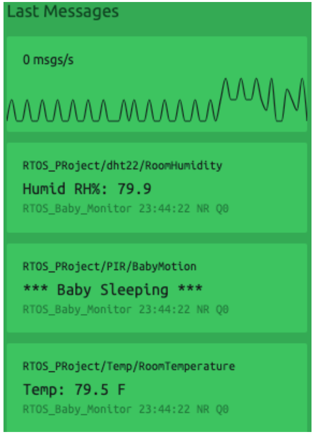
When no motion present

3. Kept the device in different room conditions so systems updated the exact temperature and humidity.



```
Last Messages
0 msgs/s

RTOS_PRoject/dht22/RoomHumidity
Humid RH%: 51.1
RTOS_Baby_Monitor 23:31:02 NR Q0

RTOS_PRoject/PIR/BabyMotion
*** Baby Sleeping ***
RTOS_Baby_Monitor 23:31:00 NR Q0

RTOS_PRoject/Temp/RoomTemperature
Temp: 75.9 F
RTOS_Baby_Monitor 23:31:02 NR Q0
```

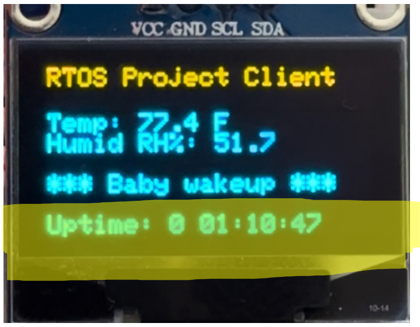
Room with low temperature and humidity



```
Last Messages
0 msgs/s

RTOS_PRoject/dht22/RoomHumidity
Humid RH%: 79.9
RTOS_Baby_Monitor 23:44:22 NR Q0

RTOS_PRoject/PIR/BabyMotion
*** Baby Sleeping ***
RTOS_Baby_Monitor 23:44:22 NR Q0

RTOS_PRoject/Temp/RoomTemperature
Temp: 79.5 F
RTOS_Baby_Monitor 23:44:22 NR Q0
```

Room with High temperature and humidity

Note: In this RTOS project using task schedular to handle two tasks in parallel even though system is up for more than an hour still works without any issue. So, this proves no issues in task implementation. See the below highlighted part from the image which shows the system uptime.



```
RTOS Project Client

Temp: 77.4 F
Humid RH%: 51.7

*** Baby wakeup ***

Uptime: 0 01:10:47
```

Limitations: As the PIR sensor is not precise it can detect the bulb light and says baby is wakeup. And other limitation, even in sleep if baby is rolled then we get baby is wakup.

Future Development:

1. I wanted to implement or add one more thread where is starts timer on detection, if PIR sensor detection triggers more that 10-15 second then only want to publish the "Baby Wakeup" Message.
2. ESP can make some buzzer sound instead LED turn on and off.
3. Also the Client response can send to Baby monitor / server device and to some task.

Conclusion : Used that Class RTOS concepts like interrupts, task scheduling, Project design, Communication Protocol and developed the RTOS Baby Monitor Project and currently system working as expected.