IoT-Based COPD Monitoring Report

Project Overview

ChronicObstructive Pulmonary Disease (COPD) is a progressive respiratory condition requiring frequent monitoring of critical vital signs, particularly oxygen saturation (SpO2) and heart rate. This project introduces a comprehensive IoT-based COPD monitoring system that leverages the MAX30102 optical sensor, RuggedBoard A5D2X for embedded processing, WE10 WiFi module for connectivity, and Rightech IoT Cloud for cloud-based monitoring, visualization, and automation.

Sensor Module - MAX30102

TheMAX30102 is an integrated pulseoximeter and heart-rate monitoring sensor developed by Maxim Integrated. It functions using the photoplethysmography (PPG) technique, where red (660 nm) and infrared (880 nm) LEDs illuminate human tissue, and a photodetector captures the reflected light. The sensor calculates:

- SpO2 using the absorption ratio of red and IR light.
- Heart Rate (BPM) using variations in the absorption due to blood pulsations.

The sensor is compact, power-efficient, and communicates via the I2C interface. It also includes ambient light cancellation and an onboard temperature sensor for enhanced accuracy.

Embedded System - RuggedBoard A5D2X

The RuggedBoard A5D2X is aLinux-powered SingleBoard Computer (SBC) based on the Microchip SAMA5D27 ARM Cortex-A5 processor. Designed for industrial and edge computing, it provides robust processing capabilities and a wide range of peripheral interfaces including I2C and UART, making it suitable for real-time health monitoring applications.

The board reads SpO2 and heart rate data from the MAX30102 sensor over I2C and formats the readings for transmission. Its ability to run full-featured Linux enables edge processing, local filtering of noise/artifacts, and even basic analytics if required.

Wireless Transmission - WE10 WiFi Module

TheWE10is a UART-configurableWiFimodule supporting IEEE 802.11 b/g/n and protocols like HTTP and MQTT. In this project, it is used to wirelessly transmit sensor data from the RuggedBoard to the cloud.

The module is programmed using AT commands and supports secure communication using WPA/WPA2 encryption. It is ideal for embedded systems requiring quick, cost-effective cloud connectivity.

Cloud Backend - Rightech IoT Cloud

Rightech IoT Cloud (RIC) is a no-code, cloud-based platform designed for rapid IoT application deployment.
It allows:

- Device modeling (virtual representation of hardware)
- Telemetry mapping and visualization
- Automation using finite state machines (FSM)
- Integration with third-party applications via REST API

Sensor data is published to the cloud over MQTT using the WE10 module. Within RIC, developers can monitor real-time values, trigger alerts (e.g., SpO2 < 90%), and log historical data for analysis.

System Integration and Data Flow

The overall data flow is structured as follows:

MAX30102 -> (I2C) -> RuggedBoard A5D2X -> (UART) -> WE10 WiFi Module -> (MQTT) -> Rightech IoT Cloud

The RuggedBoard reads data from the MAX30102, sends it via UART to the WE10 module, which transmits it over WiFi to Rightech Cloud. The cloud dashboard displays heart rate and oxygen saturation in real time and supports rule-based automation.

CODE :

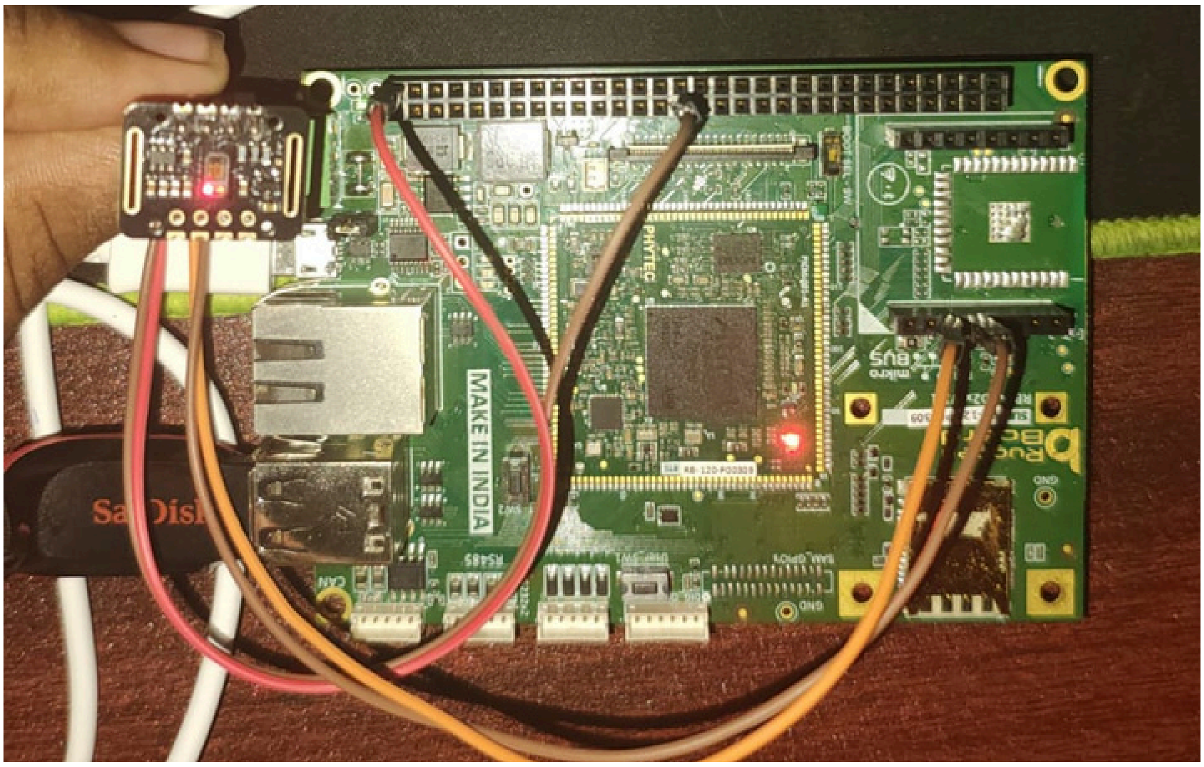
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <mraa.h>
#include <mraa/uart.h>
#include <mraa/i2c.h>
#include <sys/time.h>
#include <math.h>

#define UART_DEVICE "/dev/ttyS3"
#define MAX30102_ADDR 0x57
#define SAMPLE_SIZE 100
#define RATE_SIZE 4

mraa_uart_context uart;
mraa_i2c_context i2c;

float rates[RATE_SIZE];
int rate_index = 0;
float beat_avg = 0;
float bpm = 0;
long last_beat_time = 0;
int pulse_detected = 0;
uint32_t last_ir = 0;

void send_uart_command(const char* cmd)
{
        char buffer[256] = {0};
        mraa_uart_write(uart, cmd, strlen(cmd));
        mraa_uart_write(uart, "\r\n", 2);
        sleep(1);
        int bytes_read = mraa_uart_read(uart, buffer, sizeof(buffer));
        if (bytes_read > 0)
{
        printf("Received data: %.*s\n", bytes_read, buffer);
        }
}

void WE10_Init()
{
        /********* CMD+RESET **********/
        send_uart_command("CMD+RESET");

        /********* CMD+WIFIMODE=1 **********/
        send_uart_command("CMD+WIFIMODE=1");

        /********* CMD+CONTOAP **********/
        send_uart_command("CMD+CONTOAP=\"hiii\",\"12345678\"");

        /********* CMD?WIFI **********/
        send_uart_command("CMD?WIFI");
}

void MQTT_Init()
{
        /********* CMD+MQTTNETCFG **********/
        send_uart_command("CMD+MQTTNETCFG=dev.rightech.io,1883");

        /********* CMD+MQTTCONCFG **********/
        send_uart_command("CMD+MQTTCONCFG=3,mqtt-samantaray_asutosh-6bybv0,,,,,,,,");

        /********* CMD+MQTTSTART **********/
        send_uart_command("CMD+MQTTSTART=1");
}

int i2c_write(uint8_t reg, uint8_t data)
{
        uint8_t buf[2] = {reg, data};
        return mraa_i2c_write(i2c, buf, 2);
}

int i2c_read(uint8_t reg, uint8_t* buf, int len)
{
        mraa_i2c_write_byte(i2c, reg);
        return mraa_i2c_read(i2c, buf, len);
}

void max30102_init()
{
        i2c_write(0x09, 0x40);
usleep(100000);
        i2c_write(0x09, 0x03);
        i2c_write(0x0A, 0x27);
        i2c_write(0x0C, 0x3F);
        i2c_write(0x0D, 0x3F);
        i2c_write(0x08, 0x0F);
        i2c_write(0x02, 0x00);
        i2c_write(0x03, 0x00);
        i2c_write(0x04, 0x00);
}

void read_fifo_sample(uint32_t* red, uint32_t* ir)
{
        uint8_t buf[6];
        i2c_read(0x07, buf, 6);
        *red = ((buf[0] & 0x03) << 16) | (buf[1] << 8) | buf[2];
        *ir = ((buf[3] & 0x03) << 16) | (buf[4] << 8) | buf[5];
}

int check_for_beat(uint32_t irValue)
{
        int beat = 0;
        if (irValue > 50000 && !pulse_detected && irValue > last_ir)
{
        pulse_detected = 1;
        struct timeval now;
        gettimeofday(&now, NULL);
        long now_ms = now.tv_sec * 1000 + now.tv_usec / 1000;
        if (last_beat_time != 0)
        {
                long delta = now_ms - last_beat_time;
                bpm = 60.0 * 1000 / delta;
                if (bpm >= 40 && bpm <= 180)
                {
                        rates[rate_index++] = bpm;
                        rate_index %= RATE_SIZE;
                        beat_avg = 0;
```

```
                                for (int i = 0; i < RATE_SIZE; i++)
                                        beat_avg += rates[i];
                                        beat_avg /= RATE_SIZE;
                        }
                }
                last_beat_time = now_ms;
                beat = 1;
                }
                if (irValue < last_ir)
                pulse_detected = 0;
                last_ir = irValue;
                return beat;
}

float calculate_spo2(uint32_t* red, uint32_t* ir, int len)
{
        uint32_t red_max = 0, red_min = 0xFFFFFF;
        uint32_t ir_max = 0, ir_min = 0xFFFFFF;
        uint64_t red_sum = 0, ir_sum = 0;
        for (int i = 0; i < len; i++)
{
        if (red[i] > red_max) red_max = red[i];
        if (red[i] < red_min) red_min = red[i];
        if (ir[i] > ir_max) ir_max = ir[i];
        if (ir[i] < ir_min) ir_min = ir[i];
        red_sum += red[i];
        ir_sum += ir[i];
        }

        float red_ac = red_max - red_min;
        float red_dc = red_sum / len;
        float ir_ac = ir_max - ir_min;
        float ir_dc = ir_sum / len;

        if (red_dc < 1000 || ir_dc < 1000 || ir_ac == 0 || red_ac == 0)
        return 0;

        float r = (red_ac / red_dc) / (ir_ac / ir_dc);
        float spo2 = 110.0 - 25.0 * r;
        if (spo2 > 100) spo2 = 100;
        if (spo2 < 70) spo2 = 70;
        return spo2;
}

void publish_to_righttech(float bpm, float avg_bpm, float spo2)
{
        char topic[128];

        /********* CMD+MQTTPUB BPM **********/
        sprintf(topic, "CMD+MQTTPUB=base/health/bpm,%.1f", bpm);
        send_uart_command(topic);

        /********* CMD+MQTTPUB AVG_BPM **********/
        sprintf(topic, "CMD+MQTTPUB=base/health/avg_bpm,%.1f", avg_bpm);
        send_uart_command(topic);


        /********* CMD+MQTTPUB SpO2 **********/
        sprintf(topic, "CMD+MQTTPUB=base/health/spo2,%.1f", spo2);
        send_uart_command(topic);
}
```

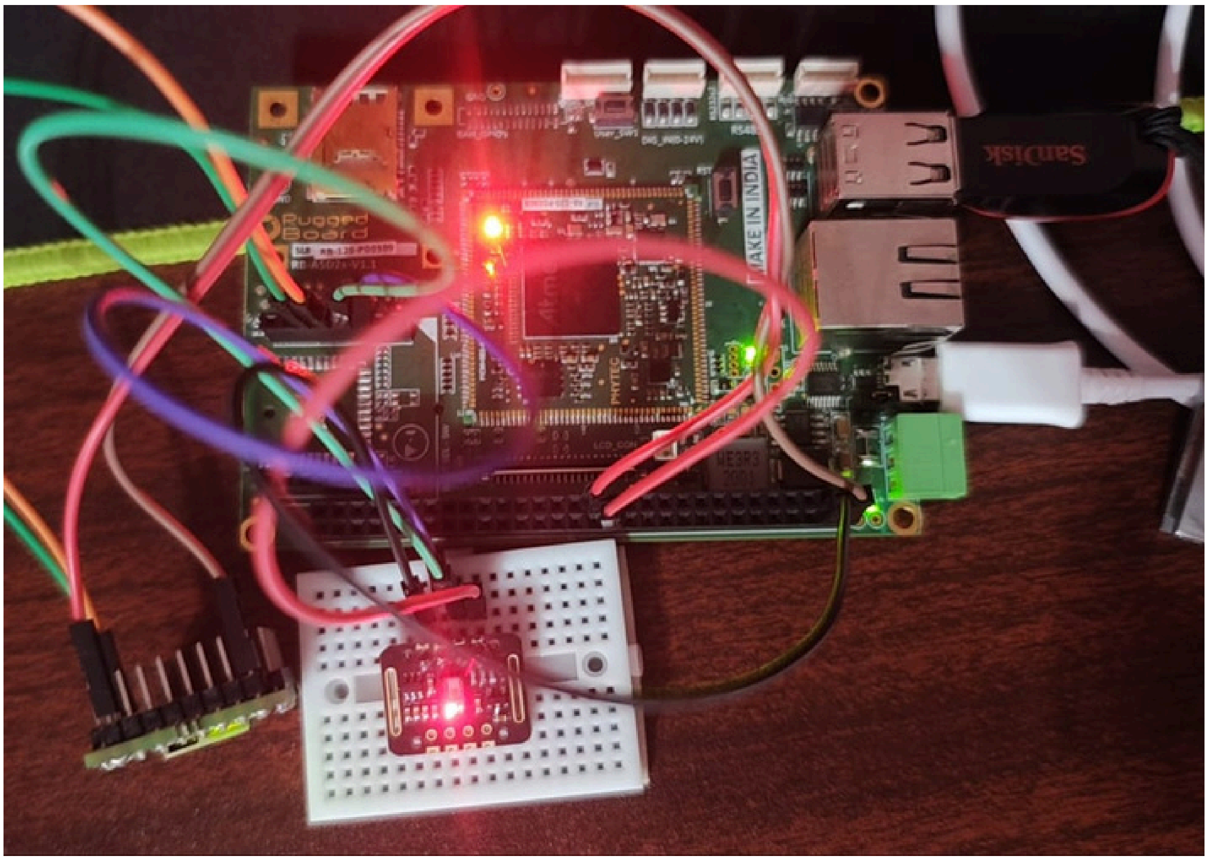
```
intmain()
{
                mraa_init();

uart= mraa_uart_init_raw(UART_DEVICE);
                if (!uart)
{
                fprintf(stderr, "Failed to open UART\n");
                return1;
                }
                mraa_uart_set_baudrate(uart,38400);
                mraa_uart_set_mode(uart,8,MRAA_UART_PARITY_NONE, 1);

                i2c = mraa_i2c_init(1);
                if (!i2c)
{
                fprintf(stderr, "Failed to init I2C\n");
                return1;
                }
                mraa_i2c_address(i2c,MAX30102_ADDR);

                WE10_Init();
                MQTT_Init();
                max30 102_init( );

                uint32_tred[SAMPLE_SIZE],ir[SAMPLE_SIZE];

                while(1)
{
                for (inti = 0;i < SAMPLE_SIZE; i++)
                 {
                                read_fifo_sample(&red[i],&ir[i]);
                                usleep(10000);
                                check_ for_beat (ir[i] );
                }

                floatspo2=calculate_spo2(red,ir,SAMPLE_SIZE);
                printf("BPM:%.1f|AvgBPM:%.1f|SpO₂: %.1f%%\n", bpm, beat_avg, spo2);

                if (bpm > 30 && spo2 > 70)
                                publish_to_righttech(bpm,beat_avg, spo2);

                sleep(1);
                }

                mraa_uart_stop(uart);
                mraa_i2c_stop(i2c);
                return0;
}
```

DEMOSTRATION:

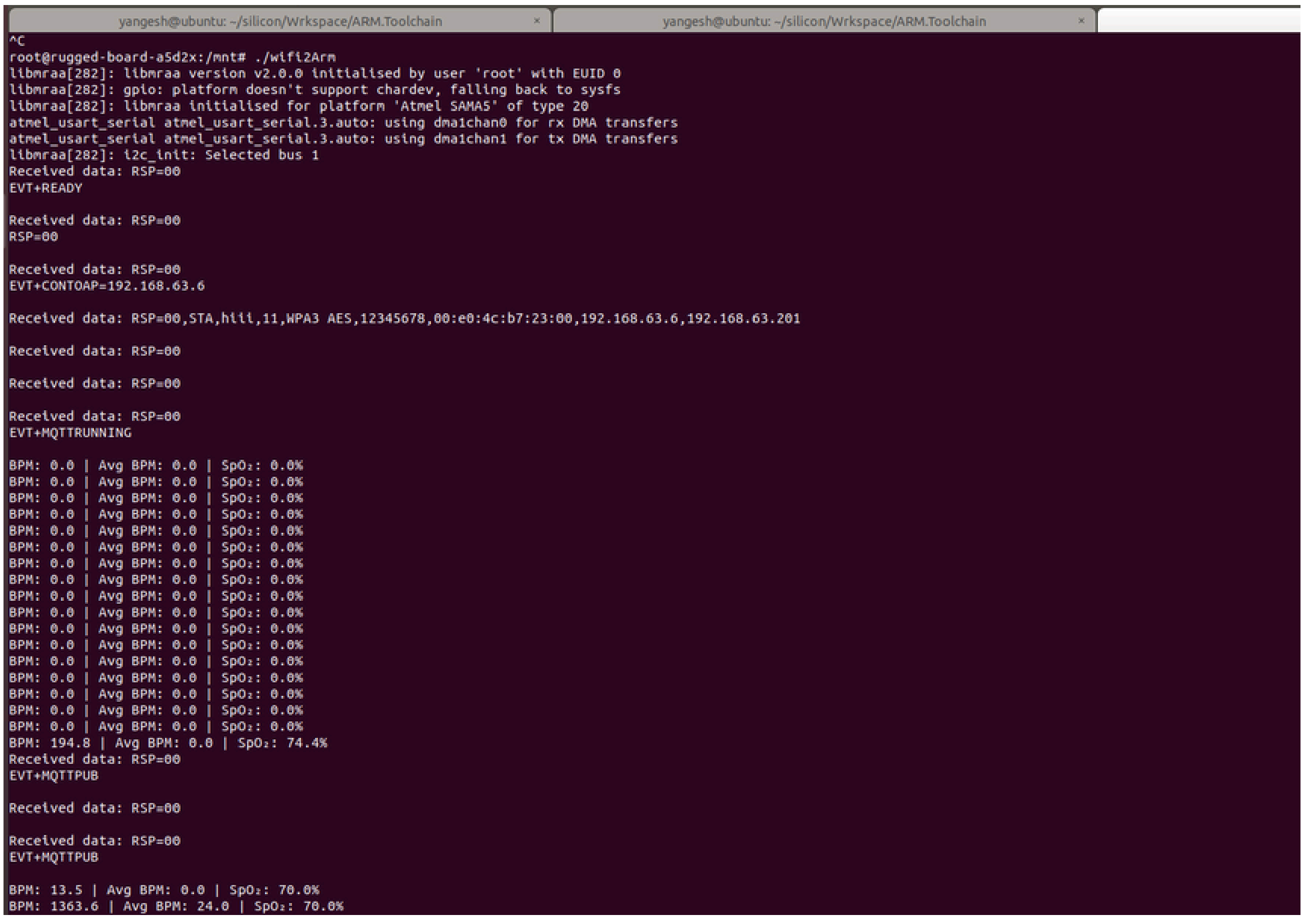
1. MAX30102 connection with Rugged Board



2. MAX30102 and W10 Wifi module connection with Rugged Board

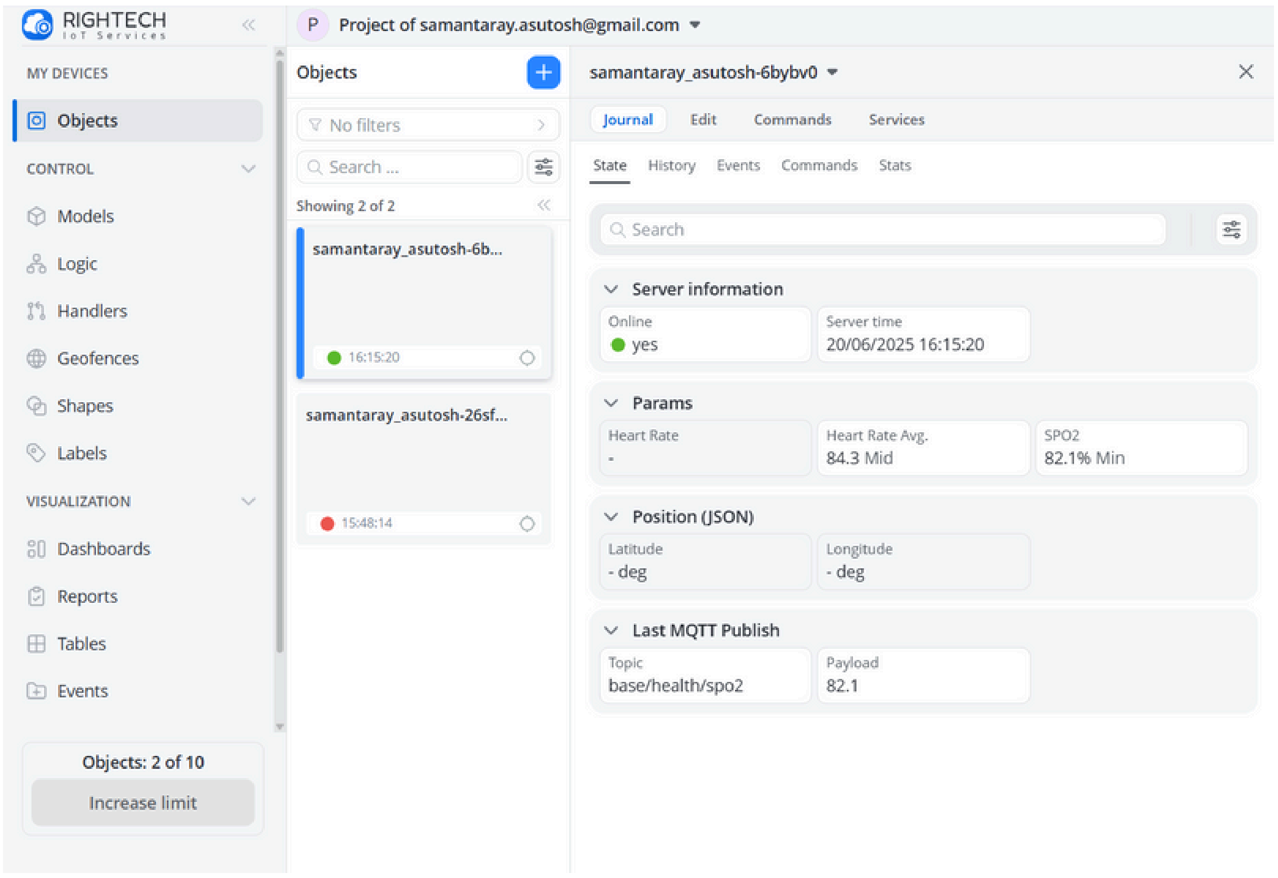


3. Execution Of ARM File in Board Terminal


yangesh@ubuntu: ~/silicon/Wrkspace/ARM.Toolchain
yangesh@ubuntu: ~/silicon/Wrkspace/ARM.Toolchain

```
^C
root@rugged-board-a5d2x:/mnt# ./wifi2Arm
libmraa[282]: libmraa version v2.0.0 initialised by user 'root' with EUID 0
libmraa[282]: gpio: platform doesn't support chardev, falling back to sysfs
libmraa[282]: libmraa initialised for platform 'Atmel SAMA5' of type 20
atmel_usart_serial atmel_usart_serial.3.auto: using dma1chan0 for rx DMA transfers
atmel_usart_serial atmel_usart_serial.3.auto: using dma1chan1 for tx DMA transfers
libmraa[282]: i2c_init: Selected bus 1
Received data: RSP=00
EVT+READY

Received data: RSP=00
RSP=00

Received data: RSP=00
EVT+CONTOAP=192.168.63.6

Received data: RSP=00,STA,hiii,11,WPA3 AES,12345678,00:e0:4c:b7:23:00,192.168.63.6,192.168.63.201

Received data: RSP=00

Received data: RSP=00

Received data: RSP=00
EVT+MQTTRUNNING

BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 0.0 | Avg BPM: 0.0 | SpO₂: 0.0%
BPM: 194.8 | Avg BPM: 0.0 | SpO₂: 74.4%
Received data: RSP=00
EVT+MQTTPUB

Received data: RSP=00

Received data: RSP=00
EVT+MQTTPUB

BPM: 13.5 | Avg BPM: 0.0 | SpO₂: 70.0%
BPM: 1363.6 | Avg BPM: 24.0 | SpO₂: 70.0%
```

4. Display of Parameters in Rightech Cloud Server by cloud transfer



P Project of samantaray.asutosh@gmail.com

Objects: 2 of 10

Showing 2 of 2

samantaray_asutosh-6b... ● 16:15:20

samantaray_asutosh-26sf... ● 15:48:14

samantaray_asutosh-6bybv0

Server information

Online: ● yes
Server time: 20/06/2025 16:15:20

Params

Heart Rate: -
Heart Rate Avg.: 84.3 Mid
SPO2: 82.1% Min

Position (JSON)

Latitude: - deg
Longitude: - deg

Last MQTT Publish

Topic: base/health/spo2
Payload: 82.1