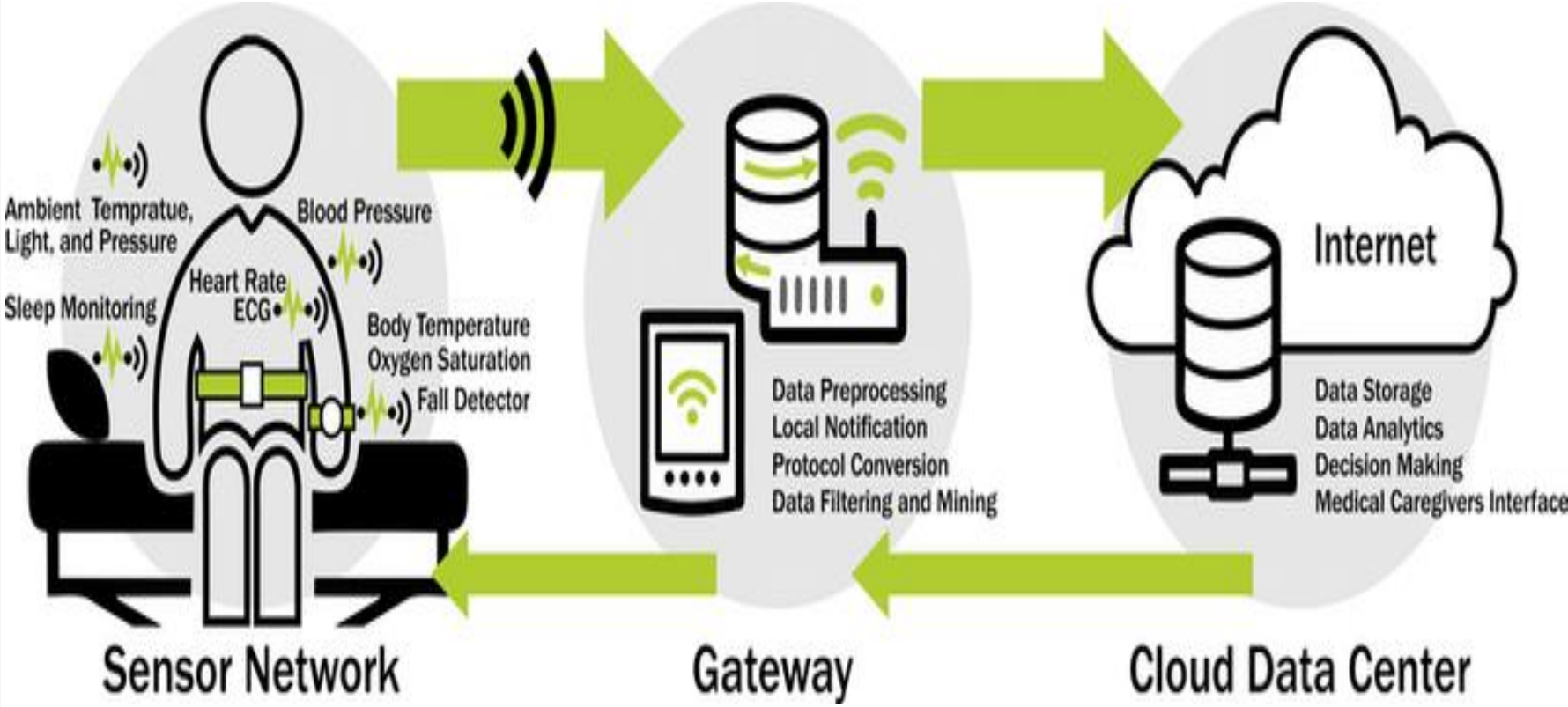
Health Monitoring System

Motivation

- Develop an efficient system that can monitor the patients' health parameters like temperature and pulse rate .
- System that can effectively deliver the data to a patient monitoring system where it is stored permanently in a database.
- Investigating the potential of Arduino and Raspberrypi in collecting, sending multiple sensed parameters in real-time.


Ambient Tempratue,
Light, and Pressure
Blood Pressure
Heart Rate
ECG
Sleep Monitoring
Body Temperature
Oxygen Saturation
Fall Detector
Data Preprocessing
Local Notification
Protocol Conversion
Data Filtering and Mining
Internet
Data Storage
Data Analytics
Decision Making
Medical Caregivers Interface
Sensor Network
Gateway
Cloud Data Center

Sensing

- This stage is responsible for reading patient's temperature and pulse rate.
- The sensors used are
  - Temperature sensor- MAX90614 IR Sensor
  - Pulse oximeter sensor- MAX30100 Pulse Oximeter
- Microcontrollers used are
  - NodeMCU
  - RaspberryPi

Networking

- Data sensed by the sensors is captured by the microcontrollers.
- This data needs to be transferred from the device to the cloud services.
- It can be done by passing MQTT messgaes to AWS IoT Services.
- It would require the setup of MQTT broker and client on RaspberryPi.

Wifi-SetUp

```
WiFiClientSecure espClient;
PubSubClient client(AWS_endpoint, 8883, callback, espClient); //set  MQTT port number to 8883
#define BUFFER_LEN 256
long lastMsg = 0;
char msg[BUFFER_LEN];
int value = 0;s
byte mac[6];
char mac_Id[18];

#define REPORTING_PERIOD_MS     1000

PulseOximeter pox;

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network

  espClient.setBufferSizes(512, 512);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  timeClient.begin();
```

Reading sensor values

```
while 1==1:
    sleep(5)
    if connflag == True:
        ethName=getEthName()
        ethMAC=getMAC(ethName)
        macIdStr = ethMAC
        Ambient_Temp = sensor.get_ambient()
        Body_Temp = sensor.get_object_1()
        paylodmsg0="{"
        paylodmsg1 = "\"mac_Id\": \""
        paylodmsg2 = "\", \"Ambient_Temp\":"
        paylodmsg3 = ", \"Body_Temp\": \""
        paylodmsg4="\"}"
        paylodmsg = "{} {} {} {} {} {} {} {}".format(paylodmsg0, pa
        paylodmsg = json.dumps(paylodmsg)
        paylodmsg_json = json.loads(paylodmsg)
        mqttc.publish("IR_TEMPERATURE", paylodmsg_json , qos=1)
        print("msg sent: IR_TEMPERATURE" ) # Print sent temperature
        print(paylodmsg_json)
```

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  pox.update();

  long now = millis();
  if (now - lastMsg > 1000) {
    lastMsg = now;

    String macIdStr = mac_Id;
    float p = pox.getHeartRate();       // Get heart rate readin
    delay(10);
    float s = pox.getSpO2();    // Get spo2 reading
    snprintf (msg, BUFFER_LEN, "{\"mac_Id\" : \"%s\", \"Pulse_R

    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("Pulse_Oximeter", msg);
    Serial.print("Heap: "); Serial.println(ESP.getFreeHeap());
  }
}
```

MQTT

- Used for sending the data from devices to AWS IoT.
- Requires client and broker connection setup.
- The clients connected to the broker sends the messages.
- The broker distributes the messages to all the devices subscribed to it.

Setting Up the Client-Broker Connection

mqttc = paho.Client()
mqttc.connect()
mqttc.publish()

```
mqttc = paho.Client()                                          # mqttc object
mqttc.on_connect = on_connect                                  # assign on_co
mqttc.on_message = on_message                                  # assign on_me
#mqttc.on_log = on_log

#### Change following parameters ####
awshost = "a2y9prqvlyko4a-ats.iot.us-east-2.amazonaws.com"      # Endpoint
awsport = 8883                                                 # Port no.
clientId = "RaspberryPi"                                        # Thing_Name
thingName = "RaspberryPi"                                       # Thing_Name
caPath = "/home/pi/AWSIoT/root-ca.pem"
certPath = "/home/pi/AWSIoT/certificate.pem.crt"
keyPath = "/home/pi/AWSIoT/private.pem.key"                                # <Th

mqttc.tls_set(caPath, certfile=certPath, keyfile=keyPath, cert_reqs=ssl.CE

mqttc.connect(awshost, awsport, keepalive=60)                  # connect to a

mqttc.loop_start()                                             # Start the lo

while 1==1:
    sleep(5)
    if connflag == True:
        ethName=getEthName()
        ethMAC=getMAC(ethName)
        macIdStr = ethMAC
        Ambient_Temp = sensor.get_ambient()
        Body_Temp = sensor.get_object_1()
        paylodmsg0="{"
        paylodmsg1 = "\"mac_Id\": \""
        paylodmsg2 = "\", \"Ambient_Temp\":"
        paylodmsg3 = ", \"Body_Temp\": \""
        paylodmsg4="\"}"
        paylodmsg = "{} {} {} {} {} {} {} {}".format(paylodmsg0, paylodmsg
        paylodmsg = json.dumps(paylodmsg)
        paylodmsg_json = json.loads(paylodmsg)
        mqttc.publish("IR_TEMPERATURE", paylodmsg_json , qos=1)        # t
        print("msg sent: IR_TEMPERATURE" ) # Print sent temperature msg on
        print(paylodmsg_json)

    else:
        print("waiting for connection...")
```

```
  File ca = SPIFFS.open("/ca.der", "r"); //replace ca eith your uploaded file name
  if (!ca) {
    Serial.println("Failed to open ca ");
  }
  else
    Serial.println("Success to open ca");

    delay(1000);

    if(espClient.loadCACert(ca))
      Serial.println("ca loaded");
    else
      Serial.println("ca failed");
      Serial.print("Heap: "); Serial.println(ESP.getFreeHeap());


    WiFi.macAddress(mac);
    snprintf(mac_Id, sizeof(mac_Id), "%02x:%02x:%02x:%02x:%02x:%02x",
    mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
    Serial.print(mac_Id);

}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  pox.update();

  long now = millis();
  if (now - lastMsg > 1000) {
    lastMsg = now;

    String macIdStr = mac_Id;
    float p = pox.getHeartRate();       // Get heart rate reading
    delay(10);
    float s = pox.getSpO2();    // Get spo2 reading
    snprintf (msg, BUFFER_LEN, "{\"mac_Id\" : \"%s\", \"Pulse_Rate\" : %f, \"Spo2\" : %f}", macIdStr.c_str(), p, s);

    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("Pulse_Oximeter", msg);
    Serial.print("Heap: "); Serial.println(ESP.getFreeHeap()); //Low heap can cause problems
  }
```

Challenges Faced

- Capturing the data simultaneously from two sensors.
- Setting up the connection between Rpi and AWS IoT.
- Displaying the data on a website.
- Setting up the wifi connection with the microcontrollers,
- Setting up the MQTT broker and Client connection.

Screenshots of the Webpage

Sensors Dashboard

Time-Stamp	Pulse_Rate	Spo2
11:57:57	59.642956	94
11:57:56	59.642956	94
11:57:55	59.642956	94
11:57:54	59.642956	94
11:57:53	59.642956	94
11:57:52	59.642956	94
11:57:51	59.642956	94
11:57:50	59.642956	94
11:57:49	59.642956	94
11:57:48	59.642956	94
11:57:47	59.642956	94
11:57:46	59.642956	94
11:57:45	59.642956	94
11:57:44	59.642956	94
11:57:43	59.642956	94
11:57:42	59.642956	94
11:57:41	59.642956	94
11:57:40	59.642956	94
11:57:39	59.642956	94
11:57:38	59.642956	94