CODE

1. Remote controlling car

```
#include "esp_camera.h"
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <iostream>
#include <sstream>

struct MOTOR_PINS
{
  int pinEn;
  int pinIN1;
  int pinIN2;
};

std::vector<MOTOR_PINS> motorPins =
{
  {12, 13, 15},  //RIGHT_MOTOR Pins (EnA, IN1, IN2)
  {12, 14, 2},  //LEFT_MOTOR  Pins (EnB, IN3, IN4)
};
#define LIGHT_PIN 4
#define UP 1
#define DOWN 2
#define LEFT 3
#define RIGHT 4
#define STOP 0

#define RIGHT_MOTOR 0
#define LEFT_MOTOR 1

#define FORWARD 1
#define BACKWARD -1

const int PWMFreq = 1000; /* 1 KHz */
const int PWMResolution = 8;
const int PWMSpeedChannel = 2;
const int PWMLightChannel = 3;

//Camera related constants
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
```

```
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

const char* ssid     = "MyWiFiCar";
const char* password = "12345678";
AsyncWebServer server(80);
AsyncWebSocket wsCamera("/Camera");
AsyncWebSocket wsCarInput("/CarInput");
uint32_t cameraClientId = 0;
const char* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(
<!DOCTYPE html>
<html>
 <head>
 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-
scalable=no">
   <style>
   .arrows {
    font-size:40px;
    color:red;
   }
   td.button {
    background-color:black;
    border-radius:25%;
    box-shadow: 5px 5px #888888;
   }
   td.button:active {
    transform: translate(5px,5px);
    box-shadow: none;
   }
   .noselect {
    -webkit-touch-callout: none; /* iOS Safari */
      -webkit-user-select: none; /* Safari */
       -khtml-user-select: none; /* Konqueror HTML */
         -moz-user-select: none; /* Firefox */
          -ms-user-select: none; /* Internet Explorer/Edge */
              user-select: none; /* Non-prefixed version, currently
                                    supported by Chrome and Opera */
   }
   .slidecontainer {
    width: 100%;
   }
   .slider {
    -webkit-appearance: none;
```

```
    width: 100%;
    height: 15px;
    border-radius: 5px;
    background: #d3d3d3;
    outline: none;
    opacity: 0.7;
    -webkit-transition: .2s;
    transition: opacity .2s;
   }
   .slider:hover {
    opacity: 1;
   }

   .slider::-webkit-slider-thumb {
    -webkit-appearance: none;
    appearance: none;
    width: 25px;
    height: 25px;
    border-radius: 50%;
    background: red;
    cursor: pointer;
   }
   .slider::-moz-range-thumb {
    width: 25px;
    height: 25px;
    border-radius: 50%;
    background: red;
    cursor: pointer;
   }
   </style>

  </head>
  <body class="noselect" align="center" style="background-color:white">

   <!--h2 style="color: teal;text-align:center;">Wi-Fi Camera &#128663; Control</h2-->

   <table id="mainTable" style="width:400px;margin:auto;table-layout:fixed" CELLSPACING=10>
    <tr>
     <img id="cameraImage" src="" style="width:400px;height:250px"></td>
    </tr>
    <tr>
     <td></td>
     <td class="button" ontouchstart='sendButtonInput("MoveCar","1")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8679;</span></td>
     <td></td>
    </tr>
    <tr>
     <td class="button" ontouchstart='sendButtonInput("MoveCar","3")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8678;</span></td>
     <td class="button"></td>
```

```
      <td class="button" ontouchstart='sendButtonInput("MoveCar","4")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8680;</span></td>
    </tr>
    <tr>
      <td></td>
      <td class="button" ontouchstart='sendButtonInput("MoveCar","2")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8681;</span></td>
      <td></td>
    </tr>
    <tr/><tr/>
    <tr>
      <td style="text-align:left"><b>Speed:</b></td>
      <td colspan=2>
       <div class="slidecontainer">
         <input type="range" min="0" max="255" value="150" class="slider" id="Speed"
oninput='sendButtonInput("Speed",value)'>
       </div>
      </td>
    </tr>
    <tr>
      <td style="text-align:left"><b>Light:</b></td>
      <td colspan=2>
       <div class="slidecontainer">
         <input type="range" min="0" max="255" value="0" class="slider" id="Light"
oninput='sendButtonInput("Light",value)'>
       </div>
      </td>
    </tr>
   </table>

   <script>
    var webSocketCameraUrl = "ws:\/\/" + window.location.hostname + "/Camera";
    var webSocketCarInputUrl = "ws:\/\/" + window.location.hostname + "/CarInput";
    var websocketCamera;
    var websocketCarInput;

    function initCameraWebSocket()
    {
     websocketCamera = new WebSocket(webSocketCameraUrl);
     websocketCamera.binaryType = 'blob';
     websocketCamera.onopen    = function(event){};
     websocketCamera.onclose   = function(event){setTimeout(initCameraWebSocket, 2000);};
     websocketCamera.onmessage = function(event)
     {
      var imageId = document.getElementById("cameraImage");
      imageId.src = URL.createObjectURL(event.data);
     };
    }
    function initCarInputWebSocket()
    {
     websocketCarInput = new WebSocket(webSocketCarInputUrl);
```

```
      websocketCarInput.onopen    = function(event)
      {
       var speedButton = document.getElementById("Speed");
       sendButtonInput("Speed", speedButton.value);
       var lightButton = document.getElementById("Light");
       sendButtonInput("Light", lightButton.value);
      };
      websocketCarInput.onclose   = function(event){setTimeout(initCarInputWebSocket, 2000);};
      websocketCarInput.onmessage = function(event){};
    }
    function initWebSocket()
    {
      initCameraWebSocket ();
      initCarInputWebSocket();
    }
    function sendButtonInput(key, value)
    {
      var data = key + "," + value;
      websocketCarInput.send(data);
    }
    window.onload = initWebSocket;
    document.getElementById("mainTable").addEventListener("touchend", function(event){
      event.preventDefault()
    });
   </script>
  </body>
</html>
)HTMLHOMEPAGE";

void rotateMotor(int motorNumber, int motorDirection)
{
  if (motorDirection == FORWARD)
  {
    digitalWrite(motorPins[motorNumber].pinIN1, HIGH);
    digitalWrite(motorPins[motorNumber].pinIN2, LOW);
  }
  else if (motorDirection == BACKWARD)
  {
    digitalWrite(motorPins[motorNumber].pinIN1, LOW);
    digitalWrite(motorPins[motorNumber].pinIN2, HIGH);
  }
  else
  {
    digitalWrite(motorPins[motorNumber].pinIN1, LOW);
    digitalWrite(motorPins[motorNumber].pinIN2, LOW);
  }
}

void moveCar(int inputValue)
{
  Serial.printf("Got value as %d\n", inputValue);
```

```
  switch(inputValue)
  {
    case UP:
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);
      break;

    case DOWN:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case LEFT:
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case RIGHT:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);
      break;

    case STOP:
      rotateMotor(RIGHT_MOTOR, STOP);
      rotateMotor(LEFT_MOTOR, STOP);
      break;

    default:
      rotateMotor(RIGHT_MOTOR, STOP);
      rotateMotor(LEFT_MOTOR, STOP);
      break;
  }
}
void handleRoot(AsyncWebServerRequest *request)
{
  request->send_P(200, "text/html", htmlHomePage);
}
void handleNotFound(AsyncWebServerRequest *request)
{
    request->send(404, "text/plain", "File Not Found");
}
void onCarInputWebSocketEvent(AsyncWebSocket *server,
                  AsyncWebSocketClient *client,
                  AwsEventType type,
                  void *arg,
                  uint8_t *data,
                  size_t len)
{
  switch (type)
  {
    case WS_EVT_CONNECT:
```

```
    Serial.printf("WebSocket client #%u connected from %s\n", client->id(), client-
>remoteIP().toString().c_str());
    break;
  case WS_EVT_DISCONNECT:
    Serial.printf("WebSocket client #%u disconnected\n", client->id());
    moveCar(0);
    ledcWrite(PWMLightChannel, 0);
    break;
  case WS_EVT_DATA:
    AwsFrameInfo *info;
    info = (AwsFrameInfo*)arg;
    if (info->final && info->index == 0 && info->len == len && info->opcode == WS_TEXT)
    {
      std::string myData = "";
      myData.assign((char *)data, len);
      std::istringstream ss(myData);
      std::string key, value;
      std::getline(ss, key, ',');
      std::getline(ss, value, ',');
      Serial.printf("Key [%s] Value[%s]\n", key.c_str(), value.c_str());
      int valueInt = atoi(value.c_str());
      if (key == "MoveCar")
      {
        moveCar(valueInt);
      }
      else if (key == "Speed")
      {
        ledcWrite(PWMSpeedChannel, valueInt);
      }
      else if (key == "Light")
      {
        ledcWrite(PWMLightChannel, valueInt);
      }
    }
    break;
  case WS_EVT_PONG:
  case WS_EVT_ERROR:
    break;
  default:
    break;
 }
}
void onCameraWebSocketEvent(AsyncWebSocket *server,
                AsyncWebSocketClient *client,
                AwsEventType type,
                void *arg,
                uint8_t *data,
                size_t len)
{
 switch (type)
 {
```

```
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u connected from %s\n", client->id(), client-
>remoteIP().toString().c_str());
      cameraClientId = client->id();
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u disconnected\n", client->id());
      cameraClientId = 0;
      break;
    case WS_EVT_DATA:
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
    default:
      break;
  }
}
void setupCamera()
{
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  config.fb_count = 1;
  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK)
  {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
```

```
  if (psramFound())
  {
    heap_caps_malloc_extmem_enable(20000);
    Serial.printf("PSRAM initialized. malloc to take memory from psram above this size");
  }
}
void sendCameraPicture()
{
  if (cameraClientId == 0)
  {
    return;
  }
  unsigned long  startTime1 = millis();
  //capture a frame
  camera_fb_t * fb = esp_camera_fb_get();
  if (!fb)
  {
     Serial.println("Frame buffer could not be acquired");
     return;
  }
  unsigned long  startTime2 = millis();
  wsCamera.binary(cameraClientId, fb->buf, fb->len);
  esp_camera_fb_return(fb);
  //Wait for message to be delivered
  while (true)
  {
    AsyncWebSocketClient * clientPointer = wsCamera.client(cameraClientId);
    if (!clientPointer || !(clientPointer->queueIsFull()))
    {
      break;
    }
    delay(1);
  }
  unsigned long  startTime3 = millis();
  Serial.printf("Time taken Total: %d|%d|%d\n",startTime3 - startTime1, startTime2 - startTime1,
startTime3-startTime2 );
}
void setUpPinModes()
{
  //Set up PWM
  ledcSetup(PWMSpeedChannel, PWMFreq, PWMResolution);
  ledcSetup(PWMLightChannel, PWMFreq, PWMResolution);

  for (int i = 0; i < motorPins.size(); i++)
  {
    pinMode(motorPins[i].pinEn, OUTPUT);
    pinMode(motorPins[i].pinIN1, OUTPUT);
    pinMode(motorPins[i].pinIN2, OUTPUT);
    /* Attach the PWM Channel to the motor enb Pin */
    ledcAttachPin(motorPins[i].pinEn, PWMSpeedChannel);
  }
```

```
  moveCar(STOP);
  pinMode(LIGHT_PIN, OUTPUT);
  ledcAttachPin(LIGHT_PIN, PWMLightChannel);
}
void setup(void)
{
  setUpPinModes();
  Serial.begin(115200);

  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.on("/", HTTP_GET, handleRoot);
  server.onNotFound(handleNotFound);

  wsCamera.onEvent(onCameraWebSocketEvent);
  server.addHandler(&wsCamera);

  wsCarInput.onEvent(onCarInputWebSocketEvent);
  server.addHandler(&wsCarInput);

  server.begin();
  Serial.println("HTTP server started");

  setupCamera();
}
void loop()
{
  wsCamera.cleanupClients();
  wsCarInput.cleanupClients();
  sendCameraPicture();
  Serial.printf("SPIRam Total heap %d, SPIRam Free Heap %d\n", ESP.getPsramSize(),
ESP.getFreePsram())}
```

2. Health Monitoring

```
#include <WiFi.h>
#include <WebServer.h>
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#include <OneWire.h>
#include <DallasTemperature.h>
#include <dht.h>

#define DHT11_PIN 18
#define DS18B20 5
#define REPORTING_PERIOD_MS     1000
```

```
float temperature, humidity, BPM, SpO2, bodytemperature;

/*Put your SSID & Password*/
const char* ssid = "Alexahome";  // Enter SSID here
const char* password = "12345678";  //Enter Password here

dht DHT;
PulseOximeter pox;
uint32_t tsLastReport = 0;
OneWire oneWire(DS18B20);
DallasTemperature sensors(&oneWire);


WebServer server(80);

void onBeatDetected()
{
 Serial.println("Beat!");
}

void setup() {
 Serial.begin(115200);
 pinMode(19, OUTPUT);
 delay(100);

 Serial.println("Connecting to ");
 Serial.println(ssid);

 //connect to your local wi-fi network
 WiFi.begin(ssid, password);

 //check wi-fi is connected to wi-fi network
 while (WiFi.status() != WL_CONNECTED) {
 delay(1000);
 Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected..!");
 Serial.print("Got IP: ");  Serial.println(WiFi.localIP());

 server.on("/", handle_OnConnect);
 server.onNotFound(handle_NotFound);

 server.begin();
 Serial.println("HTTP server started");

 Serial.print("Initializing pulse oximeter..");

 if (!pox.begin()) {
  Serial.println("FAILED");
  for (;;);
```

```
  } else {
    Serial.println("SUCCESS");
    pox.setOnBeatDetectedCallback(onBeatDetected);
  }

   pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

  // Register a callback for the beat detection

}
void loop() {
  server.handleClient();
  pox.update();
  sensors.requestTemperatures();
  int chk = DHT.read11(DHT11_PIN);

  temperature = DHT.temperature;
  humidity = DHT.humidity;
  BPM = pox.getHeartRate();
  SpO2 = pox.getSpO2();
  bodytemperature = sensors.getTempCByIndex(0);


  if (millis() - tsLastReport > REPORTING_PERIOD_MS)
  {
    Serial.print("Room Temperature: ");
    Serial.print(DHT.temperature);
    Serial.println("°C");

    Serial.print("Room Humidity: ");
    Serial.print(DHT.humidity);
    Serial.println("%");

    Serial.print("BPM: ");
    Serial.println(BPM);

    Serial.print("SpO2: ");
    Serial.print(SpO2);
    Serial.println("%");

    Serial.print("Body Temperature: ");
    Serial.print(bodytemperature);
    Serial.println("°C");

    Serial.println("*********************************");
    Serial.println();

    tsLastReport = millis();
  }

}
```

```
void handle_OnConnect() {

  server.send(200, "text/html", SendHTML(temperature, humidity, BPM, SpO2, bodytemperature));
}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}

  String SendHTML(float temperature,float humidity,float BPM,float SpO2, float bodytemperature){
  String ptr = "<!DOCTYPE html>";
  ptr +="<html>";
  ptr +="<head>";
  ptr +="<title>ESP32 Patient Health Monitoring</title>";
  ptr +="<meta name='viewport' content='width=device-width, initial-scale=1.0'>";
  ptr +="<link href='https://fonts.googleapis.com/css?family=Open+Sans:300,400,600'
rel='stylesheet'>";
  ptr +="<style>";
  ptr +="html { font-family: 'Open Sans', sans-serif; display: block; margin: 0px auto; text-align:
center;color: #444444;}";
  ptr +="body{margin: 0px;} ";
  ptr +="h1 {margin: 50px auto 30px;} ";
  ptr +=".side-by-side{display: table-cell;vertical-align: middle;position: relative;}";
  ptr +=".text{font-weight: 600;font-size: 19px;width: 200px;}";
  ptr +=".reading{font-weight: 300;font-size: 50px;padding-right: 25px;}";
  ptr +=".temperature .reading{color: #F29C1F;}";
  ptr +=".humidity .reading{color: #3B97D3;}";
  ptr +=".BPM .reading{color: #FF0000;}";
  ptr +=".SpO2 .reading{color: #955BA5;}";
  ptr +=".bodytemperature .reading{color: #F29C1F;}";
  ptr +=".superscript{font-size: 17px;font-weight: 600;position: absolute;top: 10px;}";
  ptr +=".data{padding: 10px;}";
  ptr +=".container{display: table;margin: 0 auto;}";
  ptr +=".icon{width:65px}";
  ptr +="</style>";
  ptr +="</head>";
  ptr +="<body>";
  ptr +="<h1>ESP32 Patient Health Monitoring</h1>";
  ptr +="<h3>www.how2electronics.com</h3>";
  ptr +="<div class='container'>";

  ptr +="<div class='data temperature'>";
  ptr +="<div class='side-by-side icon'>";
  ptr +="<svg enable-background='new 0 0 19.438 54.003'height=54.003px id=Layer_1 version=1.1
viewBox='0 0 19.438 54.003'width=19.438px x=0px xml:space=preserve
xmlns=http://www.w3.org/2000/svg xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path
d='M11.976,8.82v-
2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2.715,3.252,6.063v30.982";
  ptr +="C1.261,38.825,0,41.403,0,44.286c0,5.367,4.351,9.718,9.719,9.718c5.368,0,9.719-
4.351,9.719-9.718";
```

```
  ptr +="c0-2.943-1.312-5.574-3.378-7.355V18.436h-3.914v-2h3.914v-2.808h-4.084v-
2h4.084V8.82H11.976z M15.302,44.833";
  ptr +="c0,3.083-2.5,5.583-5.583,5.583s-5.583-2.5-5.583-5.583c0-2.279,1.368-4.236,3.326-
5.104V24.257C7.462,23.01,8.472,22,9.719,22";
  ptr +="s2.257,1.01,2.257,2.257V39.73C13.934,40.597,15.302,42.554,15.302,44.833z'fill=#F29C21
/></g></svg>";
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Room Temperature</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)temperature;
  ptr +="<span class='superscript'>&deg;C</span></div>";
  ptr +="</div>";

  ptr +="<div class='data humidity'>";
  ptr +="<div class='side-by-side icon'>";
  ptr +="<svg enable-background='new 0 0 29.235 40.64'height=40.64px id=Layer_1 version=1.1
viewBox='0 0 29.235 40.64'width=29.235px x=0px xml:space=preserve
xmlns=http://www.w3.org/2000/svg xmlns:xlink=http://www.w3.org/1999/xlink y=0px><path
d='M14.618,0C14.618,0,0,17.95,0,26.022C0,34.096,6.544,40.64,14.618,40.64s14.617-6.544,14.617-
14.617";
  ptr +="C29.235,17.95,14.618,0,14.618,0z M13.667,37.135c-5.604,0-10.162-4.56-10.162-10.162c0-
0.787,0.638-1.426,1.426-1.426";
  ptr
+="c0.787,0,1.425,0.639,1.425,1.426c0,4.031,3.28,7.312,7.311,7.312c0.787,0,1.425,0.638,1.425,1.4
25";
  ptr +="C15.093,36.497,14.455,37.135,13.667,37.135z'fill=#3C97D3 /></svg>";
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Room Humidity</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)humidity;
  ptr +="<span class='superscript'>%</span></div>";
  ptr +="</div>";

  ptr +="<div class='data Heart Rate'>";
  ptr +="<div class='side-by-side icon'>";
  ptr +="<svg enable-background='new 0 0 40.542 40.541'height=40.541px id=Layer_1 version=1.1
viewBox='0 0 40.542 40.541'width=40.542px x=0px xml:space=preserve
xmlns=http://www.w3.org/2000/svg xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path
d='M34.313,20.271c0-0.552,0.447-1,1-1h5.178c-0.236-4.841-2.163-9.228-5.214-12.593l-
3.425,3.424";
  ptr +="c-0.195,0.195-0.451,0.293-0.707,0.293s-0.512-0.098-0.707-0.293c-0.391-0.391-0.391-
1.023,0-1.414l3.425-3.424";
  ptr +="c-3.375-3.059-7.776-4.987-12.634-
5.215c0.015,0.067,0.041,0.13,0.041,0.202v4.687c0,0.552-0.447,1-1,1s-1-0.448-1-1V0.25";
  ptr +="c0-0.071,0.026-0.134,0.041-
0.202C14.39,0.279,9.936,2.256,6.544,5.385l3.576,3.577c0.391,0.391,0.391,1.024,0,1.414";
  ptr +="c-0.195,0.195-0.451,0.293-0.707,0.293s-0.512-0.098-0.707-0.293L5.142,6.812c-2.98,3.348-
4.858,7.682-5.092,12.459h4.804";
  ptr +="c0.552,0,1,0.448,1,1s-0.448,1-
1,1H0.05c0.525,10.728,9.362,19.271,20.22,19.271c10.857,0,19.696-8.543,20.22-19.271h-5.178";
```

```
 ptr +="C34.76,21.271,34.313,20.823,34.313,20.271z M23.084,22.037c-0.559,1.561-2.274,2.372-
3.833,1.814";
 ptr +="c-1.561-0.557-2.373-2.272-1.815-3.833c0.372-1.041,1.263-1.737,2.277-
1.928L25.2,7.202L22.497,19.05";
 ptr +="C23.196,19.843,23.464,20.973,23.084,22.037z'fill=#26B999 /></g></svg>";
 ptr +="</div>";
 ptr +="<div class='side-by-side text'>Heart Rate</div>";
 ptr +="<div class='side-by-side reading'>";
 ptr +=(int)BPM;
 ptr +="<span class='superscript'>BPM</span></div>";
 ptr +="</div>";

 ptr +="<div class='data Blood Oxygen'>";
 ptr +="<div class='side-by-side icon'>";
 ptr +="<svg enable-background='new 0 0 58.422 40.639'height=40.639px id=Layer_1 version=1.1
viewBox='0 0 58.422 40.639'width=58.422px x=0px xml:space=preserve
xmlns=http://www.w3.org/2000/svg xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path
d='M58.203,37.754l0.007-0.004L42.09,9.935l-0.001,0.001c-0.356-0.543-0.969-0.902-1.667-0.902";
 ptr +="c-0.655,0-1.231,0.32-1.595,0.808l-0.011-0.007l-0.039,0.067c-0.021,0.03-0.035,0.063-
0.054,0.094L22.78,37.692l0.008,0.004";
 ptr +="c-0.149,0.28-0.242,0.594-
0.242,0.934c0,1.102,0.894,1.995,1.994,1.995v0.015h31.888c1.101,0,1.994-0.893,1.994-1.994";
 ptr +="C58.422,38.323,58.339,38.024,58.203,37.754z'fill=#955BA5 /><path d='M19.704,38.674l-
0.013-0.004l13.544-23.522L25.13,1.156l-0.002,0.001C24.671,0.459,23.885,0,22.985,0";
 ptr +="c-0.84,0-1.582,0.41-2.051,1.038l-0.016-0.01L20.87,1.114c-0.025,0.039-0.046,0.082-
0.068,0.124L0.299,36.851l0.013,0.004";
 ptr +="C0.117,37.215,0,37.62,0,38.059c0,1.412,1.147,2.565,2.565,2.565v0.015h16.989c-0.091-
0.256-0.149-0.526-0.149-0.813";
 ptr +="C19.405,39.407,19.518,39.019,19.704,38.674z'fill=#955BA5 /></g></svg>";
 ptr +="</div>";
 ptr +="<div class='side-by-side text'>Blood Oxygen</div>";
 ptr +="<div class='side-by-side reading'>";
 ptr +=(int)SpO2;
 ptr +="<span class='superscript'>%</span></div>";
 ptr +="</div>";

 ptr +="<div class='data Body Temperature'>";
 ptr +="<div class='side-by-side icon'>";
 ptr +="<svg enable-background='new 0 0 19.438 54.003'height=54.003px id=Layer_1 version=1.1
viewBox='0 0 19.438 54.003'width=19.438px x=0px xml:space=preserve
xmlns=http://www.w3.org/2000/svg xmlns:xlink=http://www.w3.org/1999/xlink y=0px><g><path
d='M11.976,8.82v-
2h4.084V6.063C16.06,2.715,13.345,0,9.996,0H9.313C5.965,0,3.252,2.715,3.252,6.063v30.982";
 ptr +="C1.261,38.825,0,41.403,0,44.286c0,5.367,4.351,9.718,9.719,9.718c5.368,0,9.719-
4.351,9.719-9.718";
 ptr +="c0-2.943-1.312-5.574-3.378-7.355V18.436h-3.914v-2h3.914v-2.808h-4.084v-
2h4.084V8.82H11.976z M15.302,44.833";
 ptr +="c0,3.083-2.5,5.583-5.583,5.583s-5.583-2.5-5.583-5.583c0-2.279,1.368-4.236,3.326-
5.104V24.257C7.462,23.01,8.472,22,9.719,22";
 ptr +="s2.257,1.01,2.257,2.257V39.73C13.934,40.597,15.302,42.554,15.302,44.833z'fill=#F29C21
/></g></svg>";
```

```
  ptr +="</div>";
  ptr +="<div class='side-by-side text'>Body Temperature</div>";
  ptr +="<div class='side-by-side reading'>";
  ptr +=(int)bodytemperature;
  ptr +="<span class='superscript'>&deg;C</span></div>";
  ptr +="</div>";

  ptr +="</div>";
  ptr +="</body>";
  ptr +="</html>";
  return ptr;
}
```