Final Project Report

Integration of Sensor Nodes for Smart Bed in Hospitals

Master of Embedded Cyber-physical Systems

ECPS 205: Sensor Networks

University of California Irvine

Murphy Chu | 69527010
Billy Chen | 32644088
Mrunal Shailesh Sonawane | 53204178
Judit Giró Benet | 32443945

Table of Contents Page

About

As the average life expectancy grows, the importance of health care increases. Due to the rapid development of technologies, we are able to create products that can help improve medical care and help both the doctors and the patients.
This paper aims at describing the conceptual design and implementation of a smart mattress build for hospital that can help doctors monitor the condition of patients.

State of the art

The smart bed
In a world in which sensors are increasingly becoming present in most daily applications, its incorporation into the clinics is an unavoidable fact. As a great number of authors [1], [2], [3], [4] have recently pointed out, the use of sensors to monitor not only the patient's physiological constants but also the environmental conditions is currently on the rise. In fact, many authors place the start of this high pace-growing trend back to 2006, when General Electrics obtained its patents on both the "Smart bed system" [5] and the "Smart bed system and apparatus" patent [6]. The description of the smart bed included in that patent papers is the following:

"The smart bed apparatus includes a plurality of output devices […]. The smart bed computer is adapted to control one or more output devices in response to the care plan such that the patient is automatically cared for. The smart bed system includes a remotely accessible server such that one or more remotely located care givers can control output devices from a remote location in order to care for the patent." [6]

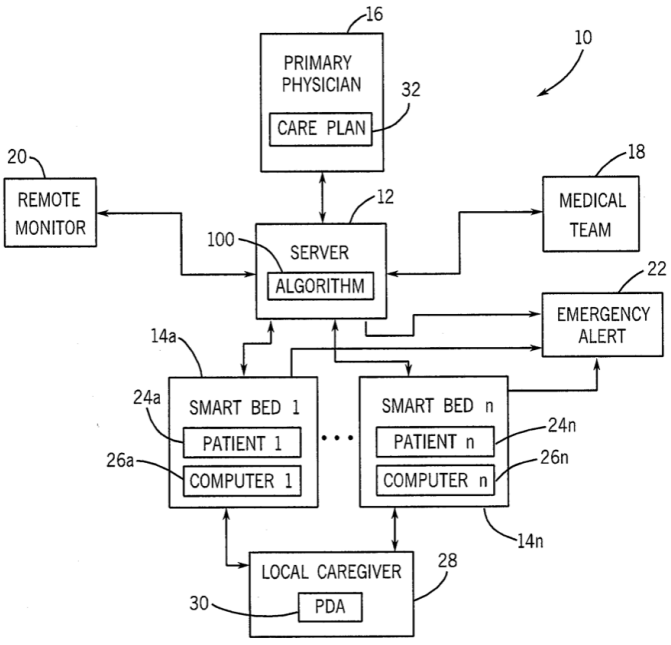

***Figure 1**. Algorithm used by General Electrics back in 2006 when they first described a smart bed [5]*

With the arrival of the smart bed concept into the clinics, a new trend started to arise. Some authors called it a "from smart health to smart hospitals" mindset, which refers to the novel disruptive idea to apply new sophisticated intelligent sensor networks to the patient. This model is based on patient physiological constants being monitored by a sensor network and output data being used to generate real-time accurate diagnosis, which has also been named a smart health concept [7].

Additionally, in the era of the P4-medicine (preventive, participatory, predictive, and personalized), the smart health concept has numerous aspects where to be applied. Example wise, the increasing volume of data that results from such a high-level monitoring health care demands for smart hospital approaches, a model that aims to tackle every decision with a specific algorithm. In this way, the medical professional can base its logical decisions on deterministic fact-based observations.

Having reviewed the main motivations for the industry to come up with the smart bed concept, the sections below will be dedicated to discussing the earlier versions of the smart bed: the smart pillow, the smart mattress and the smart sheet.

The smart pillow

The smart pillow, however, did not reach the patent office since November 2018, when Proper Pillow, Nevada patented it [8]. As described but this company, the smart pillow responds to the following:

> *A pillow is […] configured with sensors in communication with a controller having a computing component and software running in electronic memory. Using input data streams from the sensors to the controller, the software running on the computing component can ascertain a current sleep level of a user as being in a REM cycle or non REM cycle.* [8]

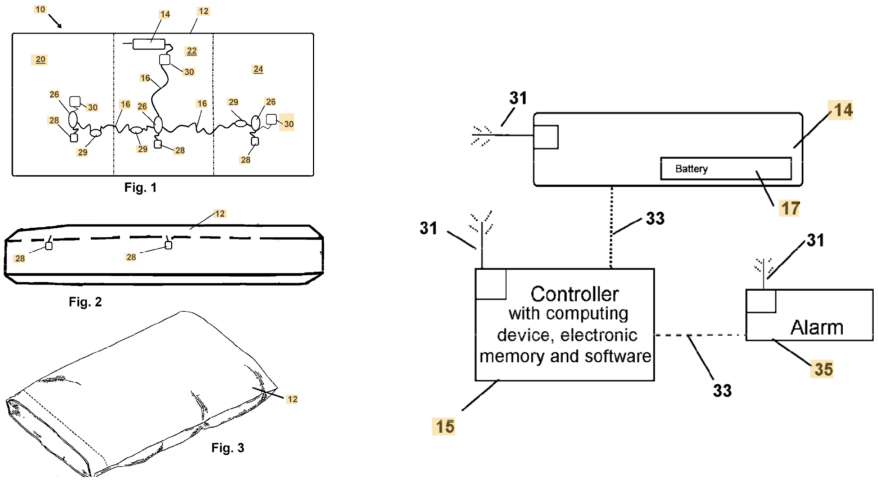

Figure 2. *Sketch of the first smart pillow published at the patent application by Proper Pillow in 2018 [5]. The sensor mat (left) is incorporated inside the pillow. The pillow controller (right) is connected to an alarm and a battery.*

In 2013 Zhang et al published a novel new technology, a smart pillow that was able to detect apnea cycles [3]. This new technology was the first time that a smart pillow was meant to be incorporated in a hospital. The goal was tackled by continuously monitoring blood oxygen levels in real-time. The system also incorporated a feedback control system to correct any possible measurement errors.

The smart sheet

The smart sheet presented by [4] is a commercial device named *SleepSmart* aimed at the monitoring of sleep for those patients presenting sleep disorders. However, its design and implementation lay close to the smart mattress that we present in this project. Similarly to the prototype presented in this project, *SleepSmart* is able to monitor sleeping position. Additionally, it is also equipped with respiration and heart rate sensors. The main idea of the model can be observed below.

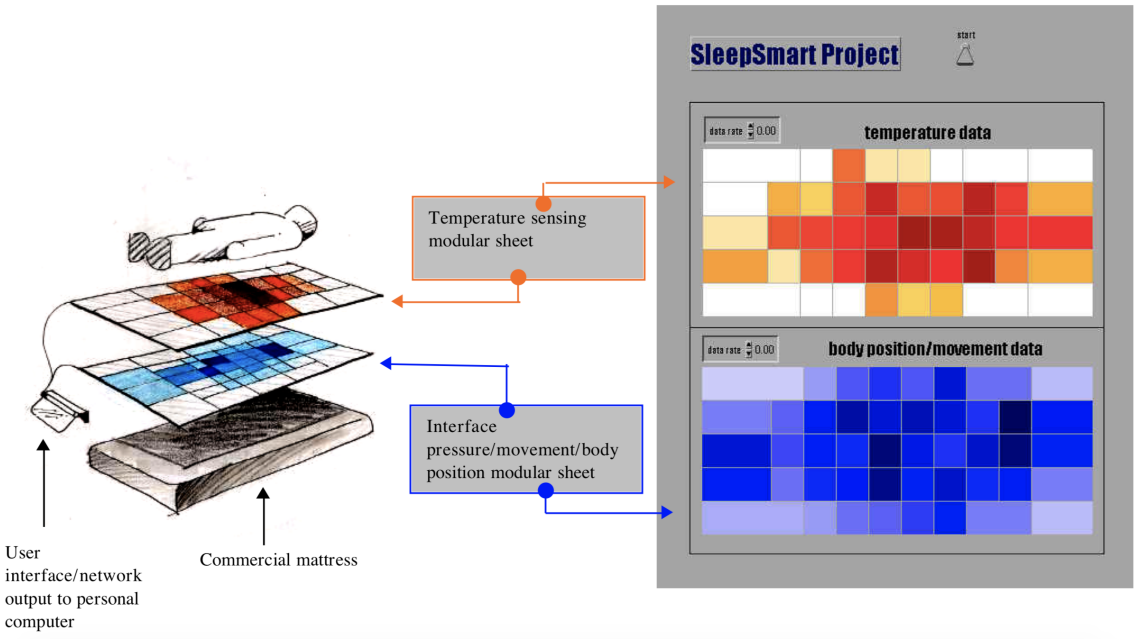

Figure 3. *Prototype concept demonstrating layered sensor approach; each layer can be updated/removed depending upon the users' needs or new technological innovations (left). Graphical user interface concept for clinical studies; pressure/temperature information is displayed chromatically and recorded for analysis (right).*

The smart mattress

One of the most recent papers on smart mattresses in that of Van der Loos et al [9], which presents *Morpheus*, a mattress actuation system that records sleeping noises (snoring) and then assists the user to change position so that the snoring decreases. Similarly to the case exposed above, this system is aimed at in-home usage for people with sleeping disorders.

After careful revision of the various published materials, it has been observed that most of the smart mattresses on the marked and the ones currently being developed are for personal use.

Hence, a marketing gap has been found, being this one the reason for the final purpose of this project.

Physical parameters to be measured

The case of Spillman et al [1] is interesting to review since it proposed a smart bed that tackles many different patient vital constants, the main goal being, that all measurements are non-invasiveness and an overall patient monitoring. In this specific case, the smart bed presented tracks respiration, heart rate and movement.

Numerous articles coincide in the utilization of fiber optic microbending based sensors for in-bed non-intrusive monitoring [1], [10]. By embedding a receiver and transmitter sensor mat inside a commercial pillow, an accuracy of ±2 beats is obtained. Furthermore, [4] ensures that "heart rate can be measured reliably if a force-sensing resistor sensor is within 2cm of the projected location".

As a conclusion, the physiological parameters that can be measured in the human body by means of a smart bed are numerous. In the examples presented by [1] and [9], patient in-bed movement and respiration thoracic movement were also measured.

System Layout

To make this project user friendly, we decided to create a control center for doctors to inspect the condition of the patient. By utilizing the Cayenne service, we created an interface for doctors to check all the data collected by sensor nodes attached to our smart mattress. The doctor can also control actuators on our nodes using the cayenne server.

By utilizing the CayenneMQTTESP32.h library, which is designed to work with Arduino, ESP8266 and ESP32 devices, we can send data to and receive data from Cayenne [11].

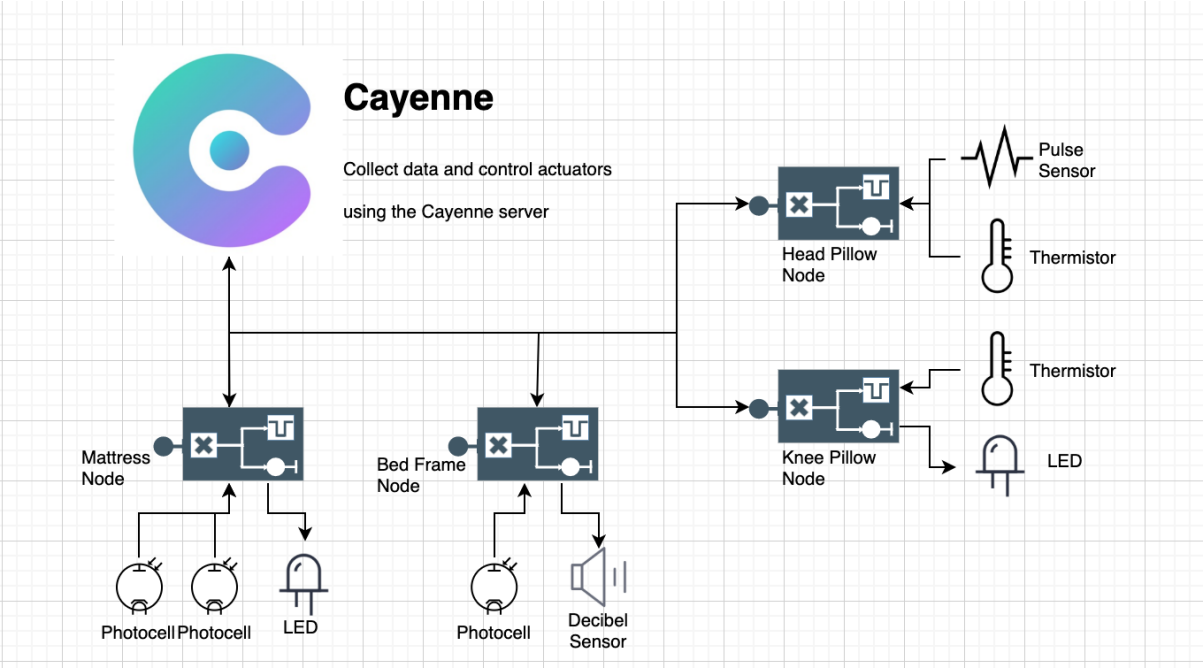

Figure 4. *Quick overview of the connectivity between the nodes of the project and Cayenne, the online server that hosts the sensor measurements in real time*

However, due to the limits of Cayenne, we were forced to separate the project into multiple servers during our demo.

Node Overview

The upcoming section is dedicated to the nodes that configure the smart bed. They will be reviewed as far as both software and hardware specifications are concerned. The sketch below shows an overview of each node's functionality as well as the interaction between them (Head pillow and Mattress nodes).

NODE 1: Mattress Light sensor 1 to detect patient lying on the right side of the bed Light sensor 2 to detect patient lying on the left side of the bed Actuator: Server sliding bar to set patient position remotely Temperature values reception from Head *Node design by Mrunal*	**NODE 2: Knee pillow** Thermistor to detect poor blood flow at patient's legs Actuator 1: If temperature beyond threshold, the server enables the doctor to send an alarm to the nurse *Node design by Billy*
NODE 3: Head pillow Pulse sensor in contact with the ear Thermistor at forehead Real-time plot to visualize heart rate and pulse Sending of temperature values to Mattress *Node design by Judit*	**NODE 4: Bed frame** Light sensor to assess if patient is resting or awake Light algorithm to assess patient condition Thermistor to assess ambient comfort *Node design by Murphy*

Figure 5. *Overview of the four nodes that integrate the smart bed. Three of the nodes contain a thermistor that captures temperature values in real time: at the forehead of the patient (3), at the legs (2) and at the ambient (4). Additionally, each node has incorporated its special features. Finally, node 1 and 3 are able to send each other UDP messages with temperature readings.*

Node 1: Head Pillow

The main idea behind the head pillow senor is the fact that the head of the patient is a part of the body where numerous measuring points to assess multiple physiological constants gather. In this specific example, two measurements were tackled.

Firstly, a pulse sensor was incorporated into the sensor node. Its readings can trigger an alarm for both an abnormal condition in the patient state (nervous, relaxed…) as well as any heart and circulatory system malfunctioning. The organization and implementation of the two sensors that form this node will be considered as follows. Figure 6 presents a diagram with the hardware components integrated in node 1.

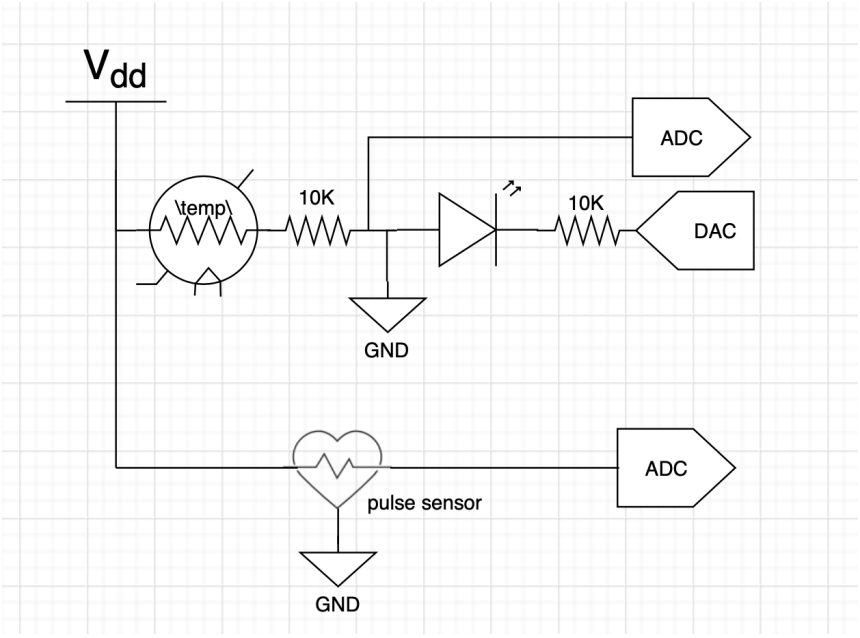

Figure 6. *Hardware components of the head pillow node. This node is composed by a thermistor (upper-right corner), an LED (upper-left corner) and a pulse sensor (lower corner).*

The embedded software of the head pillow node runs an algorithm that considers both sensors as follows.

a) **The pulse sensor**

A pulse reading is performed every 200 ms. The `PulseSensorPlayground` library [13] enables the creation of an object –Psensor in this case– that stores beat measurements. The library also provides the `.sawStartOfBeat()` function that, when applied to the `Psensor` object, returns a boolean variable indicating if the last pulse reading was a peak i.e. a beat.

b) **Thermistor**

The functionality implemented with the thermistor has already been discussed in section "Node 4: Bed frame".

Node 2: Knee Pillow

In this node, we collect the temperature of the patient, and upload it to the Cayenne server. To calculate the thermistor resistance using a formula called equation with parameter B.

$$RT = R0\, e^{B\left(\frac{1}{T} - \frac{1}{T0}\right)}$$

Expression 1. *Thermistor resistance*

Where e is the base of natural logarithm, R0 is the resistance of the thermistor measured at the temperature T0. B is a constant coefficient that depends on the characteristics of the material, it is a constant expressed in K, and its value is indicated by the manufacturers on the technical sheets.

To calculate the temperature we need to know the resistance RT using the Ohm's laws, where RT = VRT / (VR/R) [12].
By using the RT value, we can obtain the sensed temperature by the formula below:

$$T = \frac{1}{\frac{\ln(\frac{RT}{R0})}{B} + \frac{1}{T0}}$$

Expression 2. *Sensed temperature, inferred from the thermistor resistance value*

The circuit design is as below:

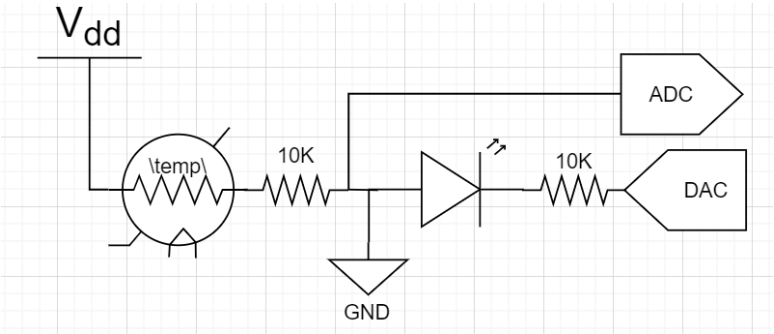

Figure 7. *Hardware components of the knee pillow node. This node is composed by a thermistor (upper-right corner), an LED (upper-left corner)*

Node 3: Mattress

The goal of this node is to obtain the sleeping position of a bed ridden patient and send alert to the caretaker to change the patient's position every 4 hours, to avoid bed sores (also known as pressure ulcers). Changing the pressure points of bedridden patients and hence allowing blood flow at the frequent pressure points in the patient's body, plays a vital role in avoiding severe bed sores.

The prototype was built using ESP32 which has the Wifi module for communication with other nodes and Cayenne (for displaying sensor data and providing control to select bed angle).
2 photoresistors used for checking the position of the patient on the mattress, they are work around substitutes for pressure sensors.
2 LEDs, one for indicating the angle of the bed's back rest ranging from 0 to 65 degrees, and the other for indicating the patient's care taker to change patient's position.

The photorestors help in determining the position of the patient by measuring the voltage across each sensor (both right and left), however, mere checking the photoresistor value should not be enough as the system can give false alarm if the sensor senses any other object apart from the patient's body. To overcome this issue, this node also considers the temperature values sent from the Pillow node, in determining the presence of a human body on the bed. Therefore, the conditions to determine the position rely on the 2 photoresistors and the temperature values received from the Pillow node using UDP.

Once the position of the patient is determined, the system senses the position for 4hrs are then sends an alert to the care taker change to a different position. If the patient is lying flat on the bed for more than 4 hours, the bed rest changes angles from 0 to 65 degrees using Cayenne to shift the pressure points from patient's back to his/her bottom.

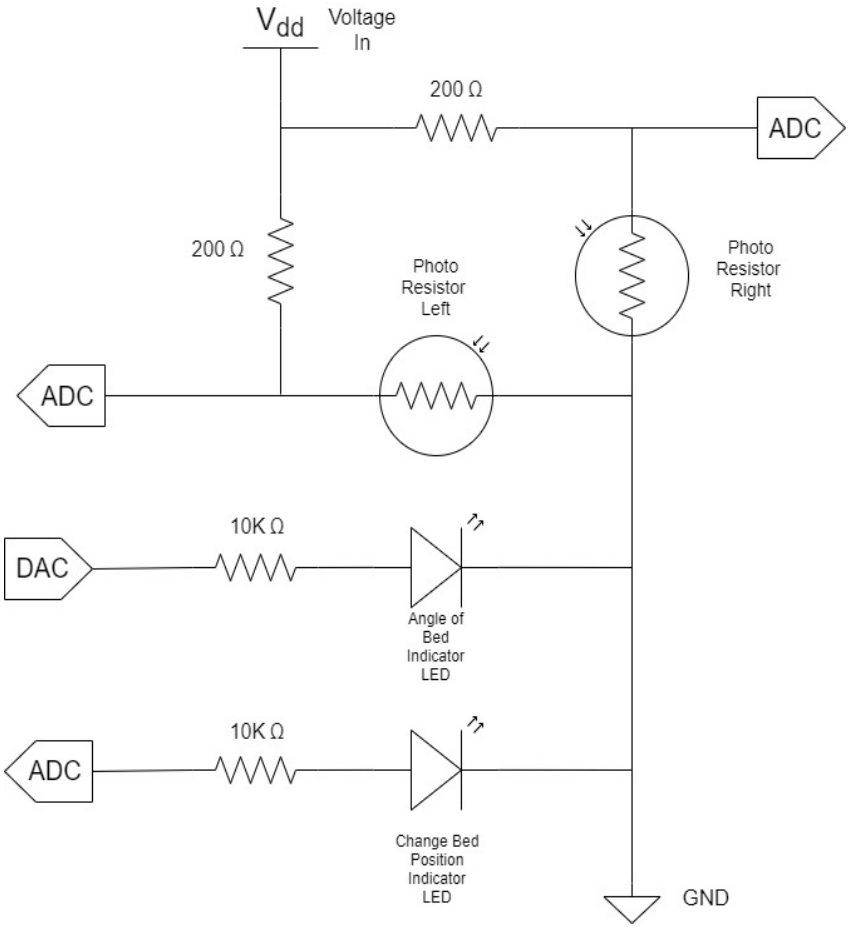

Figure 8. *Hardware components of the mattress node. This node is composed by 2 photoresistors (for right and left side), 2 indicator LEDs one for bed angle and the other for CHANGE THE POSITION Alert.*

Node 4: Bed Frame

In this node prototype, the bed frame is designed to sense the patient's room environment and conditions to see if it is optimal or acceptable for the patient to sleep and sends the average of the sensor readings to the server. Currently, the sensors that the node uses are the thermistor and the photocell. The node takes in the readings of both sensors, converts them into real world data based upon algorithms found online from adafruit, circuit basics, and arduino websites, and calculates the average light and temperature values to send the data over to Cayenne to for the user or system to evaluate. In addition to this, the node controls the LED PWM output in accordance to the

photocell's light readings. This way the maintenance check on the bed frame's circuitry can be simple to determine and easier to see the node's progress during development. At the moment, the code evaluates the average temperature and light values within its software and outputs the room's sleep environment to the serial monitor. Future development plans for this node involve the integration of a decibel(sound) sensor to check the noise/loudness level in the patient's room and the implementation of an olfactory(smell) sensor to scent the patient's condition and check for the scent of death. Additionally, the bed frame can be made with a heating or cooling function that the doctors or nurses can adjust or control based upon the patient's condition.

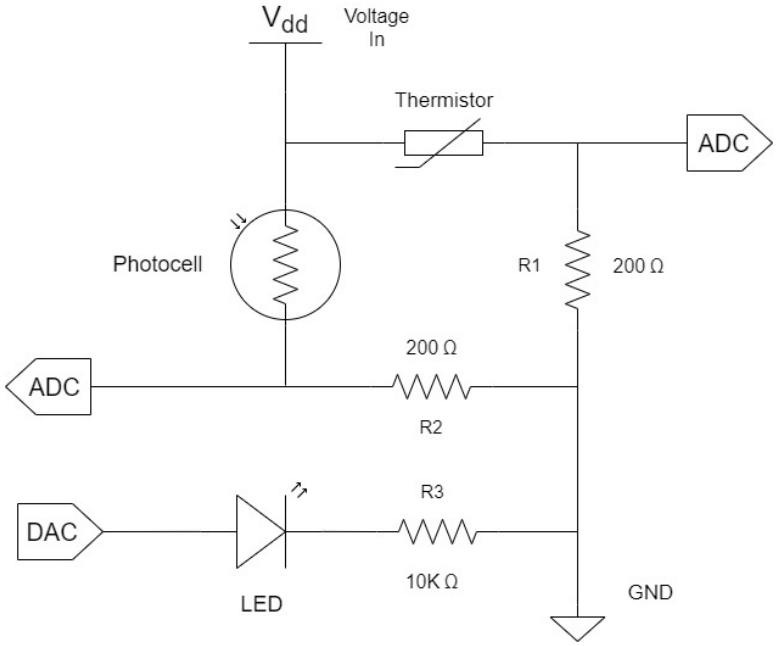

Figure 9. *Hardware components of the bed frame node. This node is composed by a thermistor (upper-right corner), and a photocell sensor (upper-left corner). An LED (lower corner) is used as a parameter indicator*

Conclusion

In conclusion, we were able to integrate different sensor nodes to sense and measure different parameters and achieve communication between the nodes, which was the primary goal of this project. Our design incorporates use of sensor nodes to obtain smart bed for hospitals to provide better health care to the patients.

After careful examination of the smart beds present in the market nowadays, we have come to the conclusion that a gap is still missing. Several different models exist, but very few of them actually succeed in integrated measurements of the overall body. The main reason behind this fact is that the medical industry is especially rigid as far as regulations are concerned. In other words, a smart bed will only be able to be launched into the market when several studies can prove that the measurements that it provides are not only accurate but also cannot pose the patient in high danger if measured wrong.

Having observed this gap in the market, the proposal that we have presented in this report aims at integrating physiological parameters of different areas, which is accomplished by presenting a smart bed that performs cardiac readings as well as temperature ones.

Finally, the main challenge encountered during the development of this project has been the real-life implementation of the project as well as the adaptation of the algorithms and the built code into real life. Some examples of this kind are that the pulse sensor readings differ depending on how the user is holding the chip. Similarly, one specific position of the patient in the bed could yield to two different readings. To solve this challenge, the algorithms have been improved to take into consideration the real-life effects of the system. In the specific case of the pulse sensor, a smoothing filter has been added to compensate for peaks when the patient stops holding the sensor right.

Code implementation

We have separated the workload into four parts. Each member was responsible for implementing one of the nodes.

Node 1: Head Pillow, done by Judit.
Node 2: Knee Pillow, done by Billy
Node 3: Mattress, done by Mrunal
Node 4: Bed Frame, done by Murphy

Please check the code below for further implementation details

Node 1: Head Pillow

This code implements some features observed in [14]

```
/*
  @file   PillowUnderHead.ino
  @brief  This is a prototype of a pillow placed under the head of patients. The goal is to
detect the body temperature of the patient.
  @author Judit
  @bug    None known
*/


/*****************************************************************************************
                        WELCOME TO THE SMARTEST "SMART BED"!
                        The Revolution happening in numerous hospitals worldwide
*****************************************************************************************/
#include <CayenneMQTTESP32.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#include <PulseSensorPlayground.h>
PulseSensorPlayground Psensor;

#define WIFI_SSID ""
#define WIFI_PASSWORD ""

int Ppin = 34;   //Analog input of pulsioximeter
int Tpin = 32;   //Analog input of pulsioximeter
int LED = 12;    //LED pin
int pulse;       //Sensor output. Range: (0,1024)
int W=10;        //Window for LP filter
int pulses[10];
int thr = 2777;  //If higher, BEAT
int BPM;         //Beats per minute

static float RT, VR, ln, Temp, Temp0, VRT, C, K, F; // For temperature calculation
char* F_char;

#define CHECKTEMPPERIOD 1500
```

```
#define RT0 10000  // Ω
#define B 3977     // K
#define VCC 3.3    //Supply voltage
#define R 10000    //R=10KΩ

void udp_send(char* data);

int T_send=millis();
int P_send=millis();
int T_temp=millis();

char CAYENNE_USERNAME[] = ""; //Username input here
char CAYENNE_PASSWORD[] = ""; //Password input here
char CAYENNE_CLIENTID[] = ""; //ClientID input here

int n=0;

/*                          SET-UP
************************************************************************/
void setup() {

  //PINOUT SET-UP
  pinMode(LED,OUTPUT);
  Serial.begin(9600);
  Cayenne.begin(CAYENNE_USERNAME,CAYENNE_PASSWORD,CAYENNE_CLIENTID,WIFI_SSID,WIFI_PASSWORD);

  //PULSE SENSOR SET-UP
  Psensor.analogInput(Ppin);
  Psensor.blinkOnPulse(LED);       //auto-magically blink Arduino's LED with heartbeat.
  Psensor.setThreshold(thr);
}



/*                          MAIN LOOP
************************************************************************/
void loop() {

 /* PULSE SENSOR READING
  ---------------------------------------*/
 pulse = analogRead(Ppin);
 Serial.println(pulse);

  if(pulse > thr){
    digitalWrite(LED,HIGH); // BEAT!
  } else {
    digitalWrite(LED,LOW);  //BACKGROUND NOISE
  }
  Serial.println(pulse);
```

```
/* PULSE READING SENT TO CAYENNE
  *  Sending every 700 ms
 ---------------------------------------*/

 if(millis() - P_send > 700){
     Cayenne.virtualWrite(10, pulse);
     P_send=millis();
 }


/* SMOOTHING FILTER
  *  Takes into account a window of [W] values
 ---------------------------------------*/
 pulses[n%30]=pulse;
 n++;
 int pulse_smooth=0;

 for(int i=0; i<W; i++)
 {
    pulse_smooth+=pulses[i];
 }
 pulse_smooth=pulse_smooth/W;
 Serial.println(pulse_smooth); //--> To be displayed at the Real-Time plot
 delay(100);


/* CHECK IF BEAT
 ---------------------------------------*/
 if(pulse > (1.2*pulse_smooth)){
   digitalWrite(LED,HIGH); // BEAT!
   Serial.println("♥  A HeartBeat Happened ! ");
   //Serial.print("BPM: ");Serial.println(BPM); // BEATS PER MINUTE

 } else {
   digitalWrite(LED,LOW);  //BACKGROUND NOISE
 }


 /* TEMPERATURE SENOR READING
 ---------------------------------------*/
  if(millis()-T_temp > 3000)
  {
      VRT = analogRead(Tpin);
      //Serial.print("T raw--> ");Serial.println(VRT);
      VRT = (5.00 / 1023.00) * VRT; // Conversion to voltage
      VR = VCC - VRT;
      RT = VRT / (VR / R); // Resistance of RT
      float tval = abs(RT/ RT0); // Added this to original code due to receiving negative values
      ln = log(tval);
      Temp = (1 / ((ln / B) + (1 / Temp0))); //Temperature from thermistor
      Temp = Temp - 273.15; //Conversion to Celsius
```

```
        C = Temp;
        K = Temp + 273.15; //Conversion to Kelvin
        F = (Temp *1.8) + 32; //Conversion to Fahrenheit

        T_temp=millis();
    }


 /* TEMPERATURE SENT TO OTHER NODE
    *  Sending every 400 ms
  ---------------------------------------*/

  if(millis() - T_send > 400)
  {
     sprintf(F_char,"%f",F);
     udp_send(F_char);
     T_send=millis();
  }
}

/* TO SEND T VALUES TO OTHER NODES
---------------------------------------*/
void udp_send(char* data){
 int i;
 Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
 Serial.println(data);
 while (data[i] != '\0') Udp.write((uint8_t)data[i++]);
 Udp.endPacket();
}
```

Node 2: Under Knee Pillow

The code presented below used features observed in references [14].

```
/*
  @file    PillowUnderKnee.ino
  @brief   This is a prototype of a pillow placed under the knee of patients. The goal is to
detect the body temperature of the patient.
  @author Billy
  @bug    Cayenne upload too slow; Improve temperature accuracy
  @Reference:   Circuit   board:   http://w...content-available-to-author-only...s.com/arduino-
thermistor-temperature-sensor-tutorial/
*/
#define THERMISPIN 36
#define LEDPIN 32
#define WIFI_SSID ""
#define WIFI_PASSWORD ""
#define CAYENNE_USERNAME ""
```

```
#define CAYENNE_PASSWORD ""
#define CAYENNE_CLIENTID ""
#define CAYENNE_PRINT Serial
#define RT0 10000   // Ω
#define B 3977      // K
#define VCC 3.3    //Supply voltage
#define R 10000  //R=10KΩ

#include <CayenneMQTTESP32.h>
#include <Ticker.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#include <math.h>

void show_temp_status(float averageTemp);

// Human body temperature references, in farenheit
const float deathHigh = 111.2;//death by overheat
const float hDeath = 109.4; //normal death
const float dhigh = 107.6; //coma
const float hhyper = 105.8; //hyperpyrexia medical emergency
const float hyperlow = 104.0; //hyperpyrexia
const float hFev = 100.9; //high fever temp/hyperthermia
const float tup = 99.5; //upper range (normal temp)
const float tlow = 97.7; //lower range (normal temp)
const float hypothm = 95.0; //hypothermia
const float ldanger = 89.6; //medical emergency
const float lcoma = 87.8; //coma
const float nearDeath = 82.4; //may appear to be dead
const float dlow = 75.2; //lower range of close to death

static WiFiUDP Udp;
static IPAddress remoteIp;
static char packetBuffer[32];

static float RT, VR, ln, Temp, Temp0, VRT, C, K, F; // For temperature calculation
static unsigned long lastCheckTemp = millis();

unsigned long lastMillis = 0;

Ticker Sensors;

void setup() {
 Serial.begin(115200);
 delay(1000);

 pinMode(LEDPIN, OUTPUT);
 // Temperature T0 from datasheet, conversion from Celsius to kelvin
 Temp0 = 25 + 273.15;
 Cayenne.begin(CAYENNE_USERNAME, CAYENNE_PASSWORD, CAYENNE_CLIENTID, WIFI_SSID, WIFI_PASSWORD);
 Sensors.attach(5, sensors);
```

```
}

void loop() {
 Cayenne.loop();
}

CAYENNE_IN(7) {
 CAYENNE_LOG("Channel %u, value %s", request.channel, getValue.asString());
 if (getValue.asInt()) {
   CAYENNE_LOG("Turn on LED!");
   digitalWrite(LEDPIN, HIGH);
 } else {
   CAYENNE_LOG("Turn off LED!");
   digitalWrite(LEDPIN, LOW);
 }
}

void show_temp_status(float averageTemp) {
 if(averageTemp >= deathHigh){
   Serial.println("Extreme Temperature: Minimal chances of survival");
 } else if(averageTemp >= hDeath){
   Serial.println("Alert!: patient needs ICU, Cardio-resporatory collapse imminent");
 } else if(averageTemp >= dhigh){
   Serial.println("Warning: patient may go into coma and might need ICU");
 } else if(averageTemp > hhyper){
   Serial.println("Alert!: Hyperpyrexia levels! Patient needs Medical Emergency!!!");
 } else if(averageTemp > hyperlow){
   Serial.println("Warning: Hyperpyrexia imminent");
 } else if(averageTemp >= hFev){
   Serial.println("Alert: High Fever/Hyperthermia!");
 } else if(averageTemp > tup){
   Serial.println("Warning: High Fever/Hyperthermia imminent");
 } else if(averageTemp >= tlow){
   Serial.println("Temperature is normal :)");
 } else if(averageTemp >= hypothm){
   Serial.println("Temperature is slightly lower than average");
 } else if(averageTemp > ldanger){
   Serial.println("Alert: Hypothermia levels!");
 } else if(averageTemp > lcoma){
   Serial.println("Alert: patient needs Medical Emergency!");
 } else if(averageTemp > nearDeath){
   Serial.println("Comatose levels");
 } else if(averageTemp >=dlow){
   Serial.println("Alert: patient needs to be in ICU heating!!!");
 } else{
   Serial.println("Extreme Low Temperature: Minimal chances of survival");
 }
}

void sensors() {
 VRT = analogRead(THERMISPIN);
```

```
 Serial.print("  -> Sensor Reading:"); Serial.println(VRT);
 VRT = (5.00 / 1023.00) * VRT; // Conversion to voltage
 VR = VCC - VRT;
 RT = VRT / (VR / R); // Resistance of RT
 float tval = abs(RT/ RT0); // Added this to original code due to receiving negative values
 ln = log(tval);
 Temp = (1 / ((ln / B) + (1 / Temp0))); //Temperature from thermistor
 Temp = Temp - 273.15; //Conversion to Celsius
 C = Temp;
 K = Temp + 273.15; //Conversion to Kelvin
 F = (Temp *1.8) + 32; //Conversion to Fahrenheit
 Serial.print("  -> Converted Temperature:"); Serial.println(F);
 show_temp_status(F);
 Cayenne.virtualWrite(5, F);
 if (F<=60.0)
       Cayenne.virtualWrite(6, 1);
 else
       Cayenne.virtualWrite(6, 0);
}
```

Node 3: Mattress

The code presented below used features observed in references [22] – [24].

```
/*
  @file   Mattress.ino
  @brief  This is a prototype of a pillow placed under the knee of patients. The goal is to
detect the body temperature of the patient.
  @author Mrunal Shailesh Sonawane
  @bug    Cayenne upload too slow;
*/
#include <CayenneMQTTESP32.h>
#include <WiFi.h>
#include <WiFiUdp.h>
#include <Ticker.h>
#include <math.h>
#define VIRTUAL_CHANNEL 1

WiFiUDP Udp;
unsigned int UDPPort = 8484;

const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;
const int controlledMotor=22;
const int photoResistorRight = 34;
```

```
const int photoResistorLeft = 35;
int sensor, controlledMotor_pwm,motorPos, tempReading;
float PhotoSensorLeft,PhotoSensorRight, controlledMotor_volts;
char packetBuffer[256];

int RIGHT,LEFT,FLAT,POS;

//
Ticker Sensors,Pos,Right,Left,Flat,Temp;

//WiFi SSID and Passcode
const char* ssid     = "";
const char* WifiPassword = "";

//Cayenne Setup
char username[] = "";
char password[] = "";
char clientID[] = "";

void sensors()
{

   //Serial.print("Sensor = ");Serial.println(sensor);
   //
   PhotoSensorRight = ((analogRead(34)) * 3.3 ) / (4095);
   PhotoSensorLeft = ((analogRead(35)) * 3.3 ) / (4095);
   //Thermistor=analogRead(36);
   Serial.print("Right  Sensor  voltage  =  ");Serial.print(PhotoSensorRight);Serial.println("  V
|");
   Cayenne.virtualWrite(1, PhotoSensorRight);
   Serial.print(" Left Sensor voltage = ");Serial.print(PhotoSensorLeft);Serial.println(" V |");
   Cayenne.virtualWrite(2, PhotoSensorLeft);


   controlledMotor_pwm=map(sensor,0,4095,0,255);

}


void setup() {
 // put your setup code here, to run once:
 Serial.begin(115200);
 Cayenne.begin(username,password,clientID,ssid,WifiPassword);
 // Connect to Wi-Fi network with SSID and password
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, WifiPassword);
 while (WiFi.status() != WL_CONNECTED) {
 delay(500);
 Serial.print(".");
 }
```

```
 // Print local IP address and start web server
 Serial.println("");
 Serial.println("WiFi connected.");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
 //myIP=WiFi.localIP();
 Udp.begin(UDPPort);

 pinMode(photoResistorRight,INPUT);
 pinMode(photoResistorLeft,INPUT);
 pinMode(controlledMotor,OUTPUT);
 pinMode(LED_BUILTIN,OUTPUT);

 ledcSetup(ledChannel, freq, resolution);
 ledcAttachPin(32, ledChannel);

 Sensors.attach(1,sensors );
 Pos.attach(1,pos);
 Temp.attach(1,readUDPPackets);
}

void loop() {
   Cayenne.loop();
}

void readUDPPackets(){
   Udp.read(packetBuffer,255);
   tempReading=atoi(packetBuffer);
}

void rightPos(){
 if ((PhotoSensorRight<0.03)&&(PhotoSensorLeft>0.03) && (tempReading>75) ){
   Serial.println("CHANGE THE POSITION!!!!!");
   digitalWrite(LED_BUILTIN,HIGH);
   //Cayenne.virtualWrite(3, "CHANGE POSITION : RIGHT");
 }
}
void leftPos(){
 if ((PhotoSensorRight>0.03)&&(PhotoSensorLeft<0.03)&&(tempReading>75)){
   Serial.println("CHANGE THE POSITION!!!!!");
   digitalWrite(LED_BUILTIN,HIGH);
 }
}
void flatPos(){
 if ((PhotoSensorRight<0.03)&&(PhotoSensorLeft<0.03)&& (tempReading>75)){
   Serial.println("CHANGE THE POSITION!!!!!");
   digitalWrite(LED_BUILTIN,HIGH);
   //CAYENNE_IN(4);

 }
}
```

```
CAYENNE_IN(4)
{
 int value = getValue.asInt(); // 0 to 65
 Serial.println("Slider");
 //CAYENNE_LOG("Channel %d, pin %d, value %d", VIRTUAL_CHANNEL, ACTUATOR_PIN, value);
 // Write the value received to the PWM pin. analogWrite accepts a value from 0 to 65.
 motorPos=map(value,0,65,0,255);
 ledcWrite(ledChannel,motorPos );
}
/*CAYENNE_IN_DEFAULT(){
    int value = getValue.asInt();
    motorPos=map(value,0,65,0,255);
    ledcWrite(ledChannel,motorPos );
}*/
void pos(){

   if ((PhotoSensorRight<0.03)&&(PhotoSensorLeft>0.03))
   {
     Serial.println ("POSTION : RIGHT");
     //Cayenne.virtualWrite(3, "POSITION : RIGHT");
     //CAYENNE_LOG("POSITION : RIGHT");
     LEFT=0;FLAT=0;digitalWrite(LED_BUILTIN,LOW);
     Left.detach();Flat.detach();
     RIGHT++;
     Serial.println(RIGHT%4 );
     if ((RIGHT%4)==0){
       Serial.println("CHANGE THE POSITION!!!!!");
       digitalWrite(LED_BUILTIN,HIGH);
       //Cayenne.virtualWrite(3, "CHANGE POSITION : RIGHT");
       RIGHT=0;
       Right.attach(1,rightPos);

       }
     }

   if ((PhotoSensorRight>0.03)&&(PhotoSensorLeft<0.03))
   {
     Serial.println ("POSTION : LEFT");
     RIGHT=0;FLAT=0;
     digitalWrite(LED_BUILTIN,LOW);
     Right.detach();Flat.detach();
     LEFT++;
     Serial.println(LEFT%4 );
     if ((LEFT%4)==0){
       Serial.println("CHANGE THE POSITION!!!!!");
       digitalWrite(LED_BUILTIN,HIGH);
       LEFT=0;
       Left.attach(1,leftPos);

       }
   }
```

```
  if ((PhotoSensorRight>0.03)&&(PhotoSensorLeft>0.03))
  {
    Serial.println ("PATIENT OUT OF PLACE");
    RIGHT=0;LEFT=0;FLAT=0;digitalWrite(LED_BUILTIN,LOW);
    Right.detach();Left.detach();Flat.detach();

  }
  if ((PhotoSensorRight<0.03)&&(PhotoSensorLeft<0.03))
  {
    Serial.println ("POSTION : FLAT");
    RIGHT=0;LEFT=0;
    Right.detach();Left.detach();digitalWrite(LED_BUILTIN,LOW);
    FLAT++;
    Serial.println(FLAT%4 );
    if ((FLAT%4)==0){
      Serial.println("CHANGE THE POSITION!!!!!");
      digitalWrite(LED_BUILTIN,HIGH);
      FLAT=0;
      Flat.attach(1,flatPos);

      }
  }
}
```

Node 4: Bed Frame

The code presented below used features observed in references [15] – [21].

```
/* Node 4: Bed Frame prototype
 *  - photocell reading (determine if room lights are on/off)
 *  - thermistor check room temp
 *
 *    - photocell and thermistor values combined should determine
 *      if environment is good for optimal sleep
 *
 *  - LED pwm brightens and darkens depending on
 *
 *  - decible sensing (loudness in room) [not provided/implemented]
 *  - smell sensing (scent of death) [not provided/implemented]
 *
 *  Cayenne Connection
 *  - Sending on Channels 8 and 9,
 *  average light and temp values respectively
 *
 * Bugs - Temperature refinement for more real time measurements
 *
 * Development notes:
 *
```

```
 * This code is done by Murphy
 *
 * Photocell reference base
 *For more information see http://l...content-available-to-author-only...t.com/photocells
 *
 * Thermistor reference bases
 *   http://w...content-available-to-author-only...s.com/arduino-thermistor-temperature-sensor-tutorial/
 *   https://c...content-available-to-author-only...o.cc/projecthub/Marcazzan_M/how-easy-is-it-to-use-a-thermistor-e39321
 *
*/
#include <CayenneMQTTESP32.h>
#include <Ticker.h>

//Ticker
Ticker Photocell;
Ticker Thermistor;
Ticker Sleep; //FIXME future note: last two channels need to be fleshed out to
Ticker Alert; //notify doc/nurse if patient should be sleeping in the room or
if room is not optimal to sleep in

//Cayenne Setup
char username[] = ""; //FIXME: Change this as needed
char password[] = "";
char clientID[] = "";

//WiFi SSID and Passcode
const char* ssid     = ""; //FIXME: change these two as needed
const char* WifiPassword = "";


#define photocellPin 35     // the cell and 10K pulldown are connected to a0
int photocellReading;     // the analog reading from the sensor divider
#define LEDpin 32          // connect Red LED to  (PWM pin)
int LEDbrightness;        //

#define thermPin 34 //thermistor pin
#define Sz 60 //(5 min for practical purposes, 30sec for demo purposes)

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;


// thermister variables
//These values are in the datasheet
#define RT0 10000   // Ω
#define B 3977      // K
//--------------------------------------
```

```
#define VCC 3.3    //Supply voltage
#define R 10000  //R=10KΩ

float RT, VR, ln, Temp, Temp0, VRT, C, K, F;


//Average store Array variables
float tsenArr[Sz];
float lsenArr[Sz];
int pos = 0;

int count = 0;

float lavg = 0.0, tavg = 0.0;


//constraint variables
//Temperature
//for adults
#define tupF 67.0 //Farenheight
#define tlowF 60.0
#define tupC  19.4444    //Celcius
#define tlowC 15.5556
#define tupK  292.594  //Kelvin
#define tlowK 288.706

//for tottlers 65 - 70 F

#define frzing 50.0
#define ovrht 90.0



//Light         //FIXME: change as  needed since photocells are not really uniform
#define lhigh 250.0 //light sleep light
#define llow 100.0 //long sleep light
#define lbest 10.0 //ideal sleep light


void setup(void) {
 //turn off built in LED
 pinMode(2, OUTPUT);
 pinMode(2, LOW);

 //Attach ticker
 Photocell.attach(1,sendToCayenne);  //FIXME  future  note:  attach  another  channels  if  deem
necessary
 Thermistor.attach(1,sendToCayenne);

 //debugging information via the Serial monitor
 Serial.begin(115200);
```

```
 Cayenne.begin(username,password,clientID,ssid,WifiPassword);
 // configure LED PWM functionalitites
 ledcSetup(ledChannel, freq, resolution);

 // attach the channel to the GPIO to be controlled
 ledcAttachPin(LEDpin, ledChannel);


 Temp0 = 25 + 273.15;                 //Temperature T0 from datasheet, conversion from Celsius
to kelvin

 for(int i = 0; i < Sz; i++){
   tsenArr[i] = 0.0;
   lsenArr[i] = 0.0;
 }
}



void sendToCayenne(){ //FIXME future note: add channels as deemed necessary
 Cayenne.virtualWrite(8, lavg); //will send values after delay time
 Cayenne.virtualWrite(9, tavg);
}


void loop(void) {
 Cayenne.loop();
 //photocell
 photocellReading = analogRead(photocellPin);

 Serial.print("Analog reading = ");
 Serial.println(photocellReading);     // the raw analog reading

 lsenArr[pos] = photocellReading; //Store in array


 //Thermistor
 VRT = analogRead(thermPin);              //Acquisition analog value of VRT

 //Serial.print("read value");
 //Serial.println(VRT);
 VRT = (5.00 / 1023.00) * VRT;      //Conversion to voltage
 //Serial.println(VRT);
 //Serial.println(VCC);
 VR = VCC - VRT;
 //Serial.println(VR);
 RT = VRT / (VR / R);               //Resistance of RT
 //Serial.print("RT: ");
 //Serial.println(RT);

 //Serial.print("RT0: ");
```

```
 //Serial.println(RT0);
 float tval = abs(RT/ RT0);//added this to original code due to receiving negative values
 //Serial.println(tval);
 ln = log(tval);
 //Serial.println(ln);
 Temp = (1 / ((ln / B) + (1 / Temp0))); //Temperature from thermistor
 //Serial.println(Temp);

 Temp = Temp - 273.15;                 //Conversion to Celsius
 C = Temp;
 /*Serial.print("Temperature:");
  Serial.print("\t");
  Serial.print(Temp);
  Serial.print("C\t\t");*/
 K = Temp + 273.15;  //Conversion to Kelvin
 /*Serial.print(K);
  Serial.print("K\t\t");*/
 F = (Temp *1.8) + 32; //Conversion to Fahrenheit
 /*Serial.print(F);
  Serial.println("F");
*/
 tsenArr[pos] = F; //Store value



 //Array position
 if(pos < (Sz - 1)){
   pos++;
 }
 else{
   pos = 0;
 }


 //code for evaluating light and temp data

 //code for evaluating light and temp data
 if(count < Sz){
   //Serial.println("waiting to fill values\n");
   count++;
 }
 else{
   //get average value of light and temp
   float tsum = 0.0, lsum = 0.0;
   for(int i = 0; i < Sz; i++){
     tsum += tsenArr[i];
     lsum += lsenArr[i];
   }

   tavg = tsum/float(Sz);
   lavg = lsum/float(Sz);
```

```
   //evaluate room data  //FIXME: send to server
   if( (lavg < lbest) && ( (tavg > tlowF) && (tavg < tupF) ) ) { //if both temp and light are
ideal
     Serial.println("Ideal sleep environment! Patient should be asleep");

   }
   else if( (lavg < llow) && ( (tavg > tlowF) && (tavg < tupF) ) ) { //if temp is ideal but light
is slightly ideal
     Serial.println("Ideal nap environment! Patient might be asleep");

   }
   else{  //if only temp or light is ideal

     if(lavg <lbest){  //if light is ideal
       Serial.println("Temperature  not  optimal,  but  have  Best  ideal  sleeping  environment
lighting");

     }
     else if(lavg <llow){ //if light is slightly ideal
       Serial.println("Temperature not optimal, but have an ideal napping environment lighting");

     }
     else if(lavg <lhigh){ //if light is close to ideal
       Serial.println("Temperature not optimal, but have ok lighting");

     }

     if( (tavg > tlowF) && (tavg < tupF) ){         //if temperature is ideal
       Serial.println("Have ideal environment temperature, but not ideal lighting for sleeping");
     }
     else if(tavg < frzing){
       Serial.println("Warning: room is freezing!");
     }
     else if(tavg > ovrht){
       Serial.println("Warning: room is over heating!");
     }
     else if(tavg < tlowF){
       Serial.println("\tPlease bring heated blanket or warm up room");
     }
     else if(tavg > tupF){
       Serial.println("\tPlease turn on the AC");
     }
     else{
         //if none of the conditions are met
       Serial.println("Patient may have difficulty or is not sleeping: Please check");
     }
   }
   Serial.print("AveTemp: "); Serial.print(tavg);
```

```
    Serial.print("\tAveLight: "); Serial.println(lavg);
  }



  // This makes LED get brighter the darker it is at the sensor
  // that means we have to -invert- the reading from 0-1023 back to 1023-0
  //photocellReading = 1023 - photocellReading;

  //now we have to map 0-1023 to 0-255 since thats the range analogWrite uses
  LEDbrightness = map(photocellReading, 0, 1023, 0, 255);
  if (LEDbrightness < 0){
    LEDbrightness = 0;
  }
  else if(LEDbrightness > 255){
    LEDbrightness = 255;
  }
  ledcWrite(ledChannel, LEDbrightness);
  Serial.print("LED brightness = ");
  Serial.println(LEDbrightness);
  Serial.println("");

  delay(500);
}
```

References

[1] SpillmanJr, W.B., Mayer, M., Bennett, J., Gong, J., Meissner, K.E., Davis, B., Claus, R.O., MuelenaerJr, A.A. and Xu, X., 2004. A 'smart'bed for non-intrusive monitoring of patient physiological factors. *Measurement Science and Technology*, *15*(8), p.1614.

[2] Hao, J., Jayachandran, M., Kng, P.L., Foo, S.F., Aung, P.W.A. and Cai, Z., 2010. FBG-based smart bed system for healthcare applications. *Frontiers of Optoelectronics in China*, *3*(1), pp.78-83.

[3] Zhang, J., Zhang, Q., Wang, Y. and Qiu, C., 2013, April. A real-time auto-adjustable smart pillow system for sleep apnea detection and treatment. In *2013 ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (pp. 179-190). IEEE.

[4] van Der Loos, H., Kobayashi, H., Liu, G., Tai, Y.Y., Ford, J., Norman, J., Tabata, T. and Osada, T., 2001, June. Unobtrusive vital signs monitoring from a multisensor bed sheet. In *Proceedings of the RESNA Conf* (pp. 218-220).

[5] Warner, A.F. and Suchecki, M.T., General Electric Co, 2008. *Smart bed system*. U.S. Patent Application 11/564,049.

[6] Warner, A.F. and Suchecki, M.T., General Electric Co, 2008. *Smart bed system and apparatus*. U.S. Patent Application 11/564,051.

[7] Holzinger, A., Röcker, C. and Ziefle, M., 2015. From smart health to smart hospitals. In *Smart health* (pp. 1-20). Springer, Cham.

[8] Loos, R., 2018. *Smart Pillow*. U.S. Patent Application 15/320,221.

[9] Van Der Loos, H.M., Ullrich, N. and Kobayashi, H., 2003. Development of sensate and robotic bed technologies for vital signs monitoring and sleep quality improvement. *Autonomous Robots*, *15*(1), pp.67-79.

[10] Chen, Z., Teo, J.T., Ng, S.H. and Yim, H., 2011, February. Smart pillow for heart-rate monitoring using a fiber optic sensor. In *Optical Fibers, Sensors, and Devices for Biomedical Diagnostics and Treatment XI* (Vol. 7894, p. 789402). International Society for Optics and Photonics.

[11] Cayenne GUI. Available online at https://developers.mydevices.com/cayenne/signin/

[12] Ostrowski, M., 2018, May. Linear and nonlinear light sensors in SCC based maximum power point search algorithms. In *Photonics for Solar Energy Systems VII* (Vol. 10688, p. 1068819). International Society for Optics and Photonics.

[13] PulseSensorPlayground library. Arduino

[14] GitHub. (2016). *WorldFamousElectronics/PulseSensorPlayground*. [online] Available at: https://github.com/WorldFamousElectronics/PulseSensorPlayground/blob/master/README.md [Accessed 10 Dec. 2019].

[15] *How Easy it is to Use a Thermistor?!* Project Hub, viewed 6 December 2019, https://create.arduino.cc/projecthub/Marcazzan_M/how-easy-is-it-to-use-a-thermistor-e39321

[16] *Illuminance – Recommended Light Level*. The Engineering Toolbox, viewed 6 December 2019, https://www.engineeringtoolbox.com/light-level-rooms-d_708.html

[17] *Light, Temperature, and Your Good Night's Sleep*. Sleepscore Labs, viewed 6 December 2019, https://www.sleepscore.com/light-temperature-and-a-good-nights-sleep/

[18] *Make an Arduino Temperature Sensor (Thermistor Tutorial)*." Circuit Basics, viewed 6 December 2019, http://www.circuitbasics.com/arduino-thermistor-temperature-sensor-tutorial/

[19] *The Ideal Temperature for Sleep*. Sleep.org, viewed 6 December 2019, https://www.sleep.org/articles/temperature-for-sleep/

[20] *Photocells: CdS Cells, Photoresistors, & Light Dependent Resistors (LDR),* Adafruit, viewed 7 December 2019, https://learn.adafruit.com/photocells

[21] *Human Body Temperature.* Wikipedia, viewed 7 December 2019, https://en.wikipedia.org/wiki/Human_body_temperature

[22] 3.5 Positioning Patients in Bed 3.5 https://opentextbc.ca/clinicalskills/chapter/3-4-positioning-a-patient-in-bed/

[23] Fabric-based Pressure Sensor Array for Decubitus Ulcer Monitoring https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4606918/

[24] Send and Receive UDP string https://www.arduino.cc/en/Tutorial/WiFiSendReceiveUDPString