# **MEDS 597B Internship Final Report**

Student Name: Xiaodong Zheng

Partner: Yinping Zhou

USC ID: 8542-8044-48

Course Name: MEDS 597a

Semester Completed: Spring 2020

Submission Date: May 7, 2020

Name of Organization: USC-ANRG

Project Name: Bluetooth Mesh Sensor network

Name of Supervisor: Bhaskar Krishnamachari

Supervisor Email Address: bkrishna@usc.edu

## Introduction:

The main purpose of this project is to investigate the feasibility of implementing a sensor network to measure and monitor a group of student’s real-time status while they are listening to the class. The project contains two parts. The first part is the single node device that is capable of measuring the student’s temperature, heartbeat rate, and skin resistance value. The second part is to create a system that collects every student’s real-time data through this network, and then analyze the data in the Linux operating system. Our internship mainly focuses on the first part of the project.

## Project Goal:

### Purpose of the work:

          The main task of this project is to create a sensor network that consists of 50 nodes. Each node can measure temperature, heartbeat rate, and galvanic skin response. The system can provide fast and real-time feedback to the lecturer, and also can provide scientific data to education area researchers.

### Medical Background:

The skin temperature, heartbeat rate, and galvanic skin response sensor are connected to the node. They measure *real*-time data produced by the user and transmit the data to the server node. The heartbeat rate and skin temperature are medical data that reflect the physiological status. The galvanic skin response refers to changes in sweat gland activity that are reflective of the intensity of the emotional state, otherwise called emotional arousal. Theses index are essential and meaningful information that signify the overall state of students, and useful for the researchers to conduct further experiments.

### System design:

The system utilizes Bluetooth Mesh network as the primary communication protocol. The network consists of 50 server nodes, one client node, and one Linux operating system based computer as the computational node to output the analysis of the data.

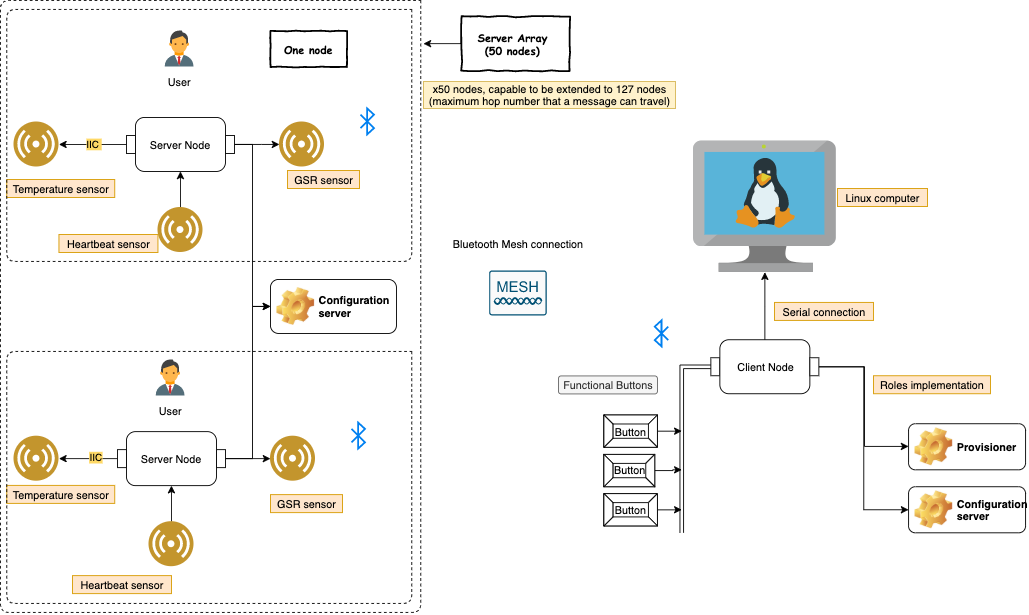


Figure 1: System overview

#### The Bluetooth mesh network:

The Bluetooth mesh network is developed based on Bluetooth low energy(BLE). It contains seven layers. In this project, only the network layer and the model layer knowledge are required. The mesh network consists of nodes sharing four common resources(details please reference mesh profile v1.1 section 3):

1. network addresses used to identify source and destination of messages
2. network keys used to secure and authenticate messages at the network layer
3. application keys used to secure and authenticate messages at the access layer
4. IV Index used to extend the lifetime of the network

In the project, the model layer application utilizes the sensor client model and sensor server model to set up the functional nodes. Also, a configuration client model is implemented in a node that is used to generate and distribute the network and application keys to make sure devices that need to communicate to each other share the proper keys for both network and access layers.

There are three roles in this mesh network:

1. The servers
2. The client
3. Provisioner.

Here are some brief explanations:

The server nodes are the device nodes that are physically connected with the sensors, and the provisioner will provision it after the network has been established. The server nodes implement two models, a sensor setup model, and a sensor server model. The two models work together to send different types of messages to the client node.

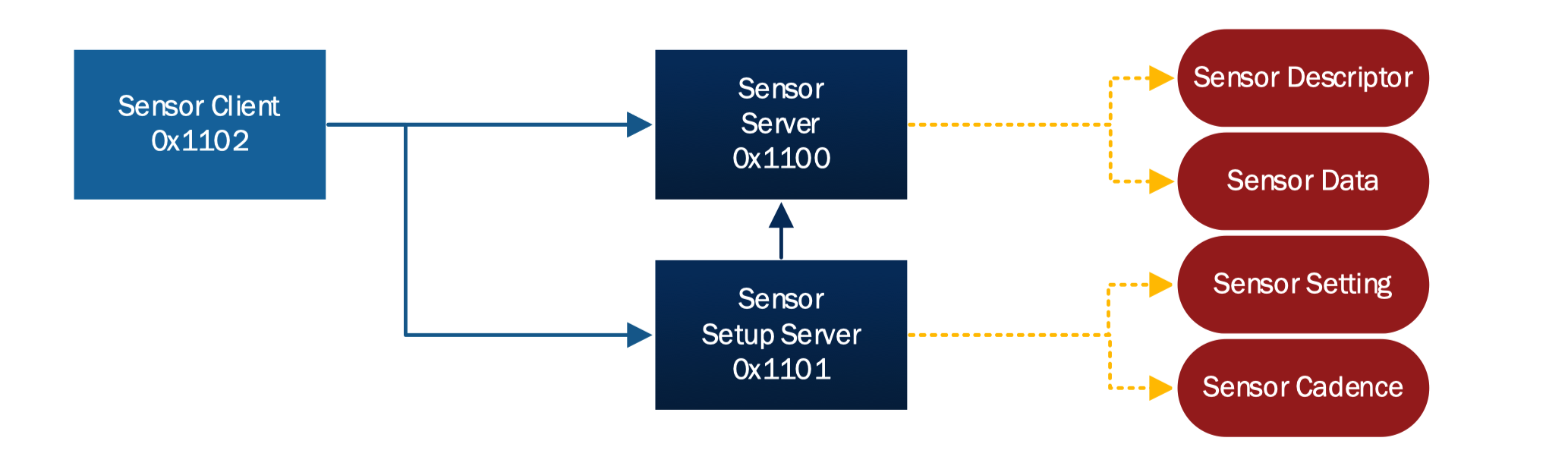
The client node is the one to send commands to and receive data from the server nodes. The client node implements a sensor client model. In the primary design of the network, there is only one client node that exists in the network. Currently, the client node is designed to have an external button to send the commands to the server nodes. Every time the button has been pressed, the corresponding types of messages will be sent to the server node and receive the immediate replies. 

Figure 2: Illustration of Sensor server model and sensor client model

*(reference: model specification 4.5 )*

The provisioner is the creator and the organizer of the network. Its primary responsibilities are to create the Bluetooth mesh network and provision device nodes that want to join into the network by distributing the addresses(but the application keys and network keys are distributed by configuration client model as mentioned above). The provisioner is currently implemented in the client node as well as the configuration client model for convenience. However, they can also be isolated from the client node.

The computer:

To achieve a higher computational capability, a Linux operating system based computer has been introduced into the network. There are basically two ways to connect the computer to the network. The first one is to connect the computer to the client node through a serial port to read the printed data received by the client. We can directly print the data from the terminal to a .csv file by using Linux command( for example, “tee” command), or using third party software that converts the data from the command line tools.

For example, to print output from the monitoring result from command line to a .csv file,

**idf.py monitor -p [port number] | tee ouput.csv**

\*\*(idf.py is the ESP IDF framework development tool, it can monitor the log information of the connected device)

The second way to connect the computer to the network is to convert the computer to a mesh node, connecting to the network via Bluetooth. However, currently Linux operating system is not officially supported by the BLE mesh stack, the BLE mesh network only compatible with device supports BLE core specification 4.0 and higher version. However, a permissible way to achieve this is to use BlueZ on Raspberry Pi(Linux Platform). The BlueZ 5.1 supports Raspiberry Pi as a Bluetooth mesh node and allow it to join Bluetooth mesh network. Here is the instructions of [how to deploy BlueZ to Raspberry Pi 4.](https://www.bluetooth.com/blog/use-bluez-v5-50-andraspberry-pi-4-updateto-create-a-bluetoothmesh-provisioner/)

#### The node implementing SoC(System on Chip):

This project uses ESP32-DevKitC as the mainboard to implementing the functionalities of the nodes. The ESP32-DevKitC is produced by Espressif, and empowered by ESP-IDF framework to development BLE mesh applications. It has a high performance and the complete Bluetooth mesh stack implementation. Also, the low price is the shining point of it, an ordinary ESP32 DevKit is only approximate $10. Here is the ESP-IDF [GitHub repository address.](https://github.com/espressif/esp-idf)

#### The sensors:

Temperature sensor: OTI-301, this temperature sensor is an infrared sensor, the specification about the sensor: [Specification](https://drive.google.com/file/d/1mmdkHkNbwC5VgQxGMrq07XolajNvgBkN/view?usp=sharing)

The sensor uses IIC bus to communicate with the server board via GPIO 18 and GPIO 19, and it has a high accuracy of ± 0.3C. When the sensor server node sends a get command, it replies with 6 bytes data, the first three bytes are ambient temperature data, and the rest are object temperature data. Here is the signal sequence diagram and formula to get the temperature data:

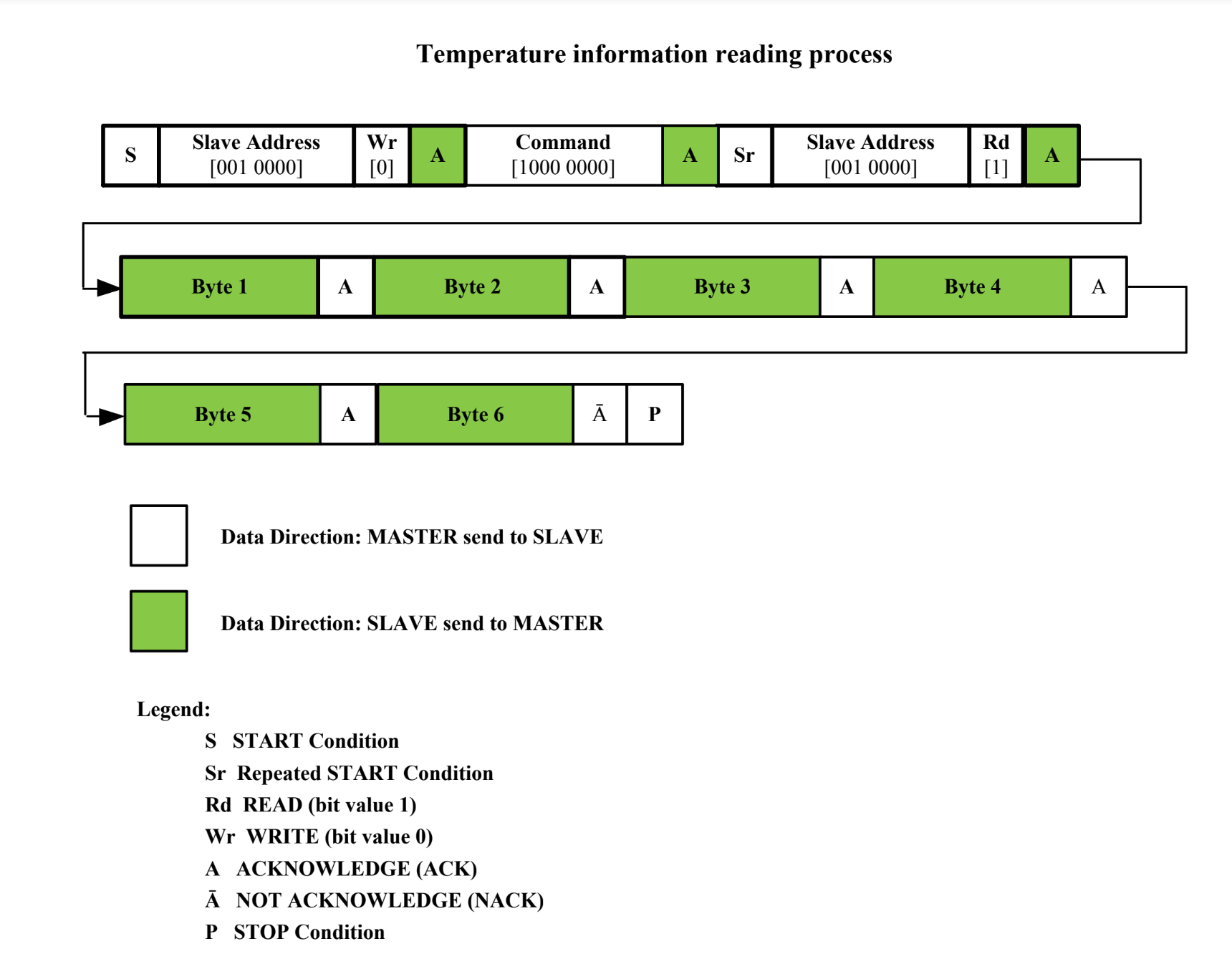
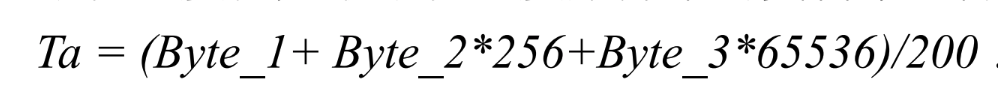
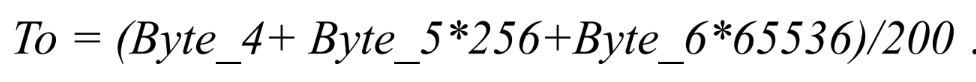


Figure 3: Temperature reading process



Equation1: Ambient temperature formula



Equation 2: Object temperature reading process

And the schematic with ESP32

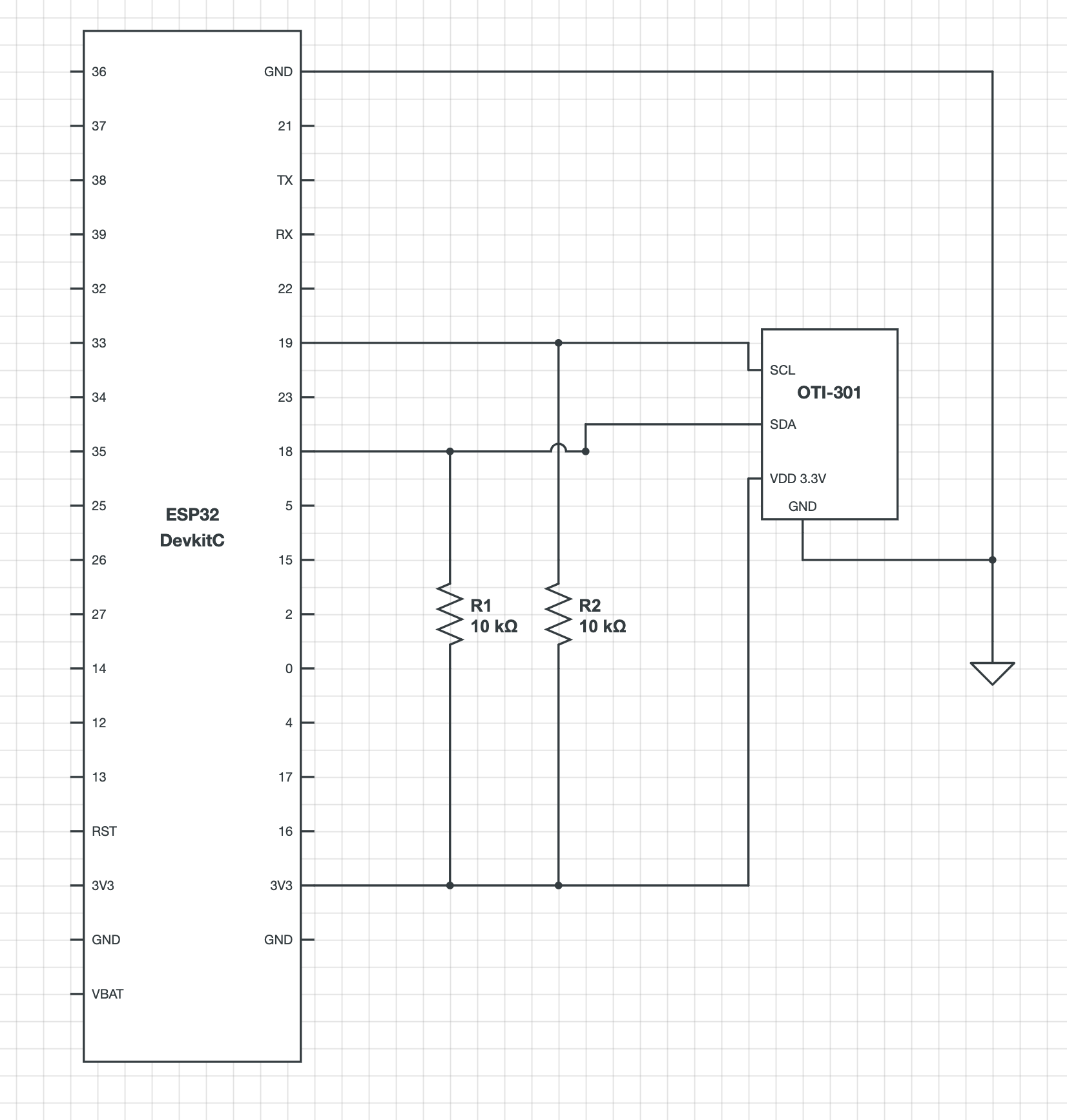


Figure 4: ESP 32 DevKit C to OTI-301

Alternatively, a contact temperature could also be used. The sensor DS18B20 is a one-wire waterproof temperature sensor. Here is the [library](https://github.com/nimaltd/ds18b20) of this sensor. The sensor DS18B20 has high resolution and accuracy, however, it also has the deflect for most contact temperature, a slow responding time. It usually takes 40 to 60 seconds to reach a stable readings since the temperature data is acquired by its steel probe, and the heat conduction takes time to complete.

#### Other sensors (Still in development):

The usages of other sensors are similar to the temperature sensor. The heartbeat sensor and the GSR sensor also use one-wire data line to transmit data, and the GSR sensor comes with two finger-cots, which makes it easy to use. The model of Heartbeat sensor is [Genuine Pulse Sensor](https://www.amazon.com/Tragoods-Monitor-Sensor-Arduino-Pulsesensor/dp/B07GGNNLJL/ref=sr_1_1?dchild=1&keywords=heartbeat+Pulse+Sensor&qid=1588902502&s=electronics&sr=1-1), and the model of GSR sensor is [Grove GSR Sensor Module 3.3V/5V](https://www.amazon.com/seeed-studio-Seeedstudio-Grove-sensor/dp/B012TNYDE4/ref=sr_1_1?dchild=1&keywords=GSR+sensor&qid=1588902431&s=electronics&sr=1-1).

#### Step by Step instruction to reproduce:

1. Install ESP-IDF prerequisite for Ubuntu

sudo apt-get install git wget flex bison gperf python python-pip python-setuptools cmake ninja-buicld ccache libffi-dev libssl-dev dfu-util

1. Clone the ESP-IDF to local folder

mkdir esp

cd esp

clone <https://github.com/espressif/esp-idf.git>

1. Install the toolchains:

cd ~/esp/esp-idf

./install.sh

1. Set up environment variables

. $HOME/esp/esp-idf/export.sh

1. Clone the project: clone https://github.com/ANRGUSC/ble\_sensor\_mesh.git
2. Connect the devices and sensors as mentioned above
3. Connect one ESP board to the computer
4. Run /dev/tty to check your port name
5. Run idf.py menuconfig in the esp folder to configure the flash parameters
6. Run idf.py build flash monitor -p [Your Port Number] In the directory ble\_sensor\_mesh\ble\_sensor\_mesh\_model\sensor\_client
7. When flashing the client program, press the left button on the board
8. Connect another ESP32 to computer and don’t close the terminal you have opened
9. Run idf.py build flash monitor -p [Your Port Number] In the directory ble\_sensor\_mesh\ble\_sensor\_mesh\_model\sensor\_server

After flash the project to the boards, the monitor program will be activated. The log information will be printed in the terminal. The sensor server will be automatically provisioned by the client once it has been detected, and the log information on the client side will show like below

A picture containing green, sitting, table, holding

Description automatically generated

Figure 5: Client provision successfully

A picture containing sitting, green

Description automatically generated

Figure 6: Server side provision success log information

Then you can try to get the temperature sensor information and readings by pressing the left button on the client board. The information of the sensor will show up like below:

A screenshot of a cell phone

Description automatically generated

Figure 7: Server side temperature readings

(The data is sent by multiple hexadecimal numbers as a row)

A picture containing phone, green

Description automatically generated

Figure 8: Client side received temperature data

**Discussion**:

The project is not a mature project yet, it still needs a lot of work to development. Currently, only the temperature sensor and button functionalities are finished, the heartbeat sensor and the GSR sensor are still in development. However, the framework is almost done. The BLE mesh is capable to contain 32768 nodes, and every node can relay the messages to each other. That makes it a very efficient and highly extendable network, and hence very suitable for developing a sensor network.

#### Future development:

There are many possible and exciting progress that could be made in the future. The complete system should be able to visualize the real-time data received on the Linux computer. To achieve this, we can use [Grafana](https://grafana.com/) and [InfluxDB](https://portal.influxdata.com/downloads/).

Also, the server node can even have its own screen to show the real-time data, so the user can view the data himself/herself.

Moreover, the client node can realize a button array to get the user’s data directionally or use a number pad instead to get the data from the server node by number.

**Conclusion**:

The project is very helpful for us to understand the BLE mesh stack. Through this project, we practiced a lot and gained the SoC development experience. Although the project is not complete yet, we think it has the potential to be developed as a system that has true values. In the end, great thanks to Professor Bhaskar Krishnamachari. His supports and encourages are the motive power for us to do the project.

#### Appendix:

* [Bluetooth Mesh specification V1.01](https://www.bluetooth.com/specifications/mesh-specifications/)
* [Grafana & InfluxDB](https://medium.com/@ashrafur/beginning-visualization-with-grafana-and-influxdb-81701e10569d)
* [ESP-IDF development guide](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html#introduction)

#### Material link:

* [ESP32S](https://www.ebay.com/itm/ESP32-ESP-32S-NodeMCU-Development-Board-2-4GHz-WiFi-Bluetooth-Dual-Mode-CP2102/382601606021?_trkparms=ispr%3D1&hash=item5914d53385:g:Jf0AAOSwsRVbznU~&enc=AQAEAAACYIQvEcHUrT7nmUC3yY5qbPyaBN1nJEDYW8MyypsJPgXKpYSzz9%2FsG0TJDO4uQ4aUXABwJ3XMjogbuz3kQyl8TizaOtKbz2EiItalooeDyqSI%2FLk6BCgNAYyxVp%2FVd9o%2FUoc249AgdckTBo2VbqHavB1jKXgQn6WurSLnY8H6Vh8iO9anFRanlcPtQ8roiks1qGjWqncBoiDWezL%2B1SnWV8sHqUwEUxVLBE9LNAycfcNuQhypIIuNlfDbwfaFxtFXLJ%2FLZCSpHXOy%2BCbo0XU89yrUVdiU4Ikt66QEu9dbLUkqtRQ0IJldN4btkWTCvD8lw02%2Bjl%2BVxR4jlkPCDI2QAia4C4ieCUuj%2BrGEVIBIwZRU7qIZrcVy%2BJ%2B8hZDwIdB5wOn7AUgfjbSrzhyejufr7lwqPmyVOpnpmKkXkbmcTFO6W8wj7dBJljakXRsamtyWCgO%2F2Eq3kgM0jSOZmHOw2YhZtss2SJBiAlN2M5Fl0wzk4uxSBk3mrIC2NxrcELqCm2BTmtISFyrGOLpSX8EGytPPP6G1SeuXGsLay4s7YMzohjvtba7kZk87hTJzs%2F7LW9I%2BbZZL24gaxuOKWS0dBVvCTJ104hvd8%2FAd2CKfvorXKM4p2oxBq%2FnbdK2ufyJ4lJbviJ98%2F2xU7tbXoKKOtjYz9bDy5a7KW0HtXQD7HMYx9PkRTedN2FD3Nve4evcPFhx96FPCL8h7AuBAjgyCwTSSsRh85Jt4C4qmFT5JS37lk%2B2vMEBrWjsNhVXsOVCOAfbpF3F74YAdnMwxlTdOVcFNgAT7VAz1SVVphLc1ydQM&checksum=382601606021a22602111ca94d369b2d3b1e1b62d639)(same as devkitC)
* [Maiji OTI-301 T420 D1 Non-Contact Digital Infrared Temperature Sensor Module](https://www.amazon.com/gp/product/B07V52M45H/ref=ppx_yo_dt_b_asin_title_o09_s00?ie=UTF8&psc=1)
* [DS1802B temperature sensor](https://www.amazon.com/gp/product/B07V6SJ3XR/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&th=1)
* [Heartbeat sensor](https://www.amazon.com/KY-039-Finger-Heartbeat-Measurement-Arduino/dp/B07869C43V/ref=sr_1_1?dchild=1&keywords=heartbeat+sensor&qid=1588910040&s=industrial&sr=1-1)
* [GSR sensor](https://www.amazon.com/seeed-studio-Seeedstudio-Grove-sensor/dp/B012TNYDE4/ref=sr_1_4?crid=2YLR1PHXJTZ1Z&dchild=1&keywords=galvanic+skin+response+sensor&qid=1588910211&sprefix=Galvanic+skin+response+%2Caps%2C203&sr=8-4)
* [Breadboard Kit](https://www.amazon.com/ELEGOO-Electronics-Kit-arranque-componentes/dp/B01ERPEMAC/ref=sr_1_1_sspa?dchild=1&keywords=Breadboard+kit&qid=1588909984&s=industrial&sr=1-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyT1daWUJSS041WE1NJmVuY3J5cHRlZElkPUEwOTA5NDAyMzFSNlBEVkMzRTFYSCZlbmNyeXB0ZWRBZElkPUEwODA5NDE2R1JNTDk5SENRTU1aJndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ==)