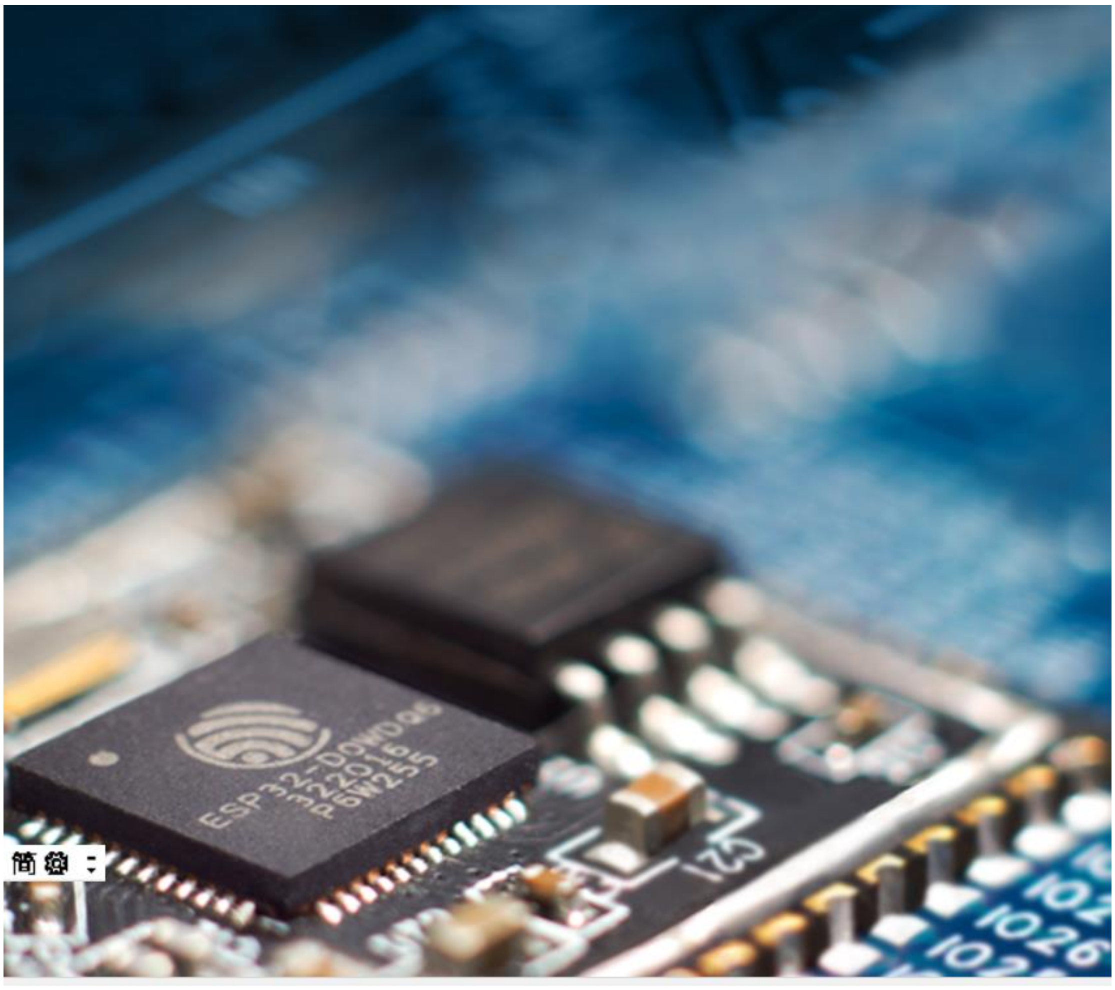
The report of Internet Of Things

XiJia Wang,ChengSi Liu,Shuo CUI

Ⅰ.Introduction

ESP32 is a series of low cost, low power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations and includes in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules.

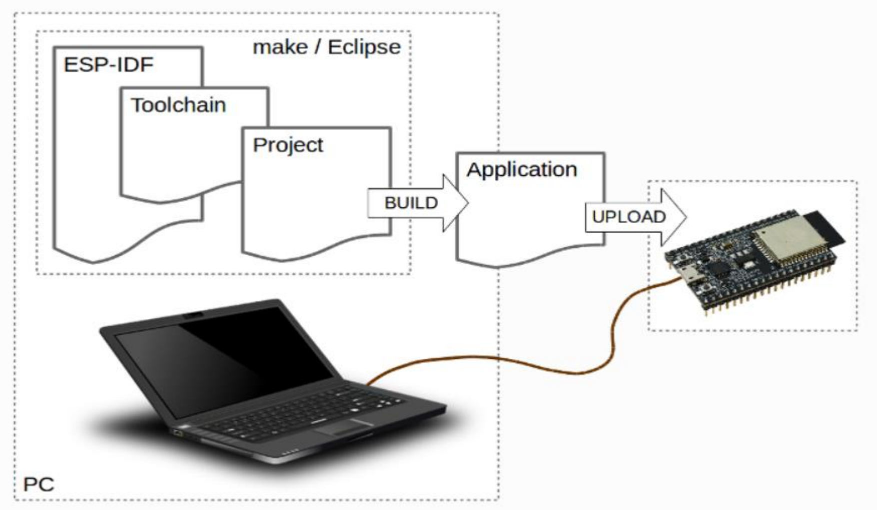
ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process.It is a successor to the ESP8266microcontroller.



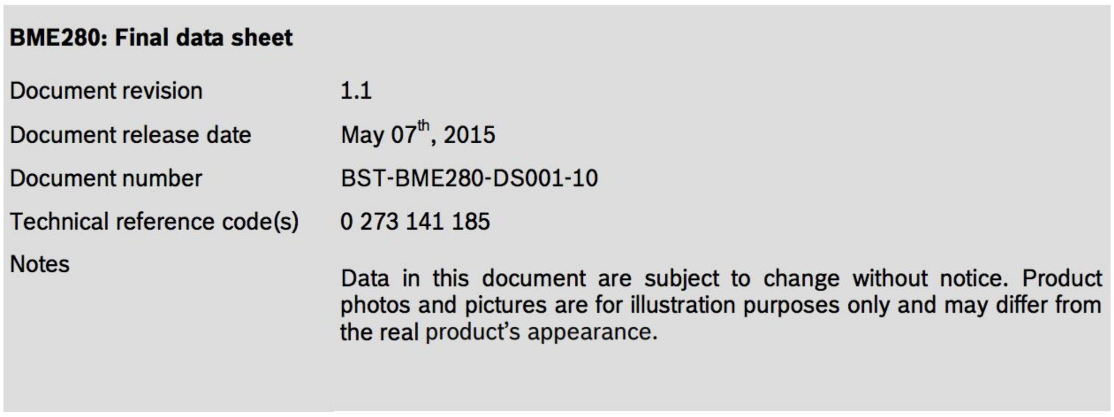
Ⅱ.prepare the workspace

ESP32 integrates Wi-Fi (2.4 GHz band) and Bluetooth 4.2 solutions on a single chip, along with dual high performance cores, Ultra Low Power co-processor and several peripherals. Powered by 40 nm technology, ESP32 provides a robust, highly integrated platform to meet the continuous demands for efficient power usage, compact design, security, high performance, and reliability.

Espressif provides the basic hardware and software resources that help application developers to build their ideas around the ESP32 series hardware. The software development framework by Espressif is intended for rapidly developing Internet-of-Things (IoT) applications, with Wi-Fi, Bluetooth, power management and several other system features.



Development of applications for ESP32



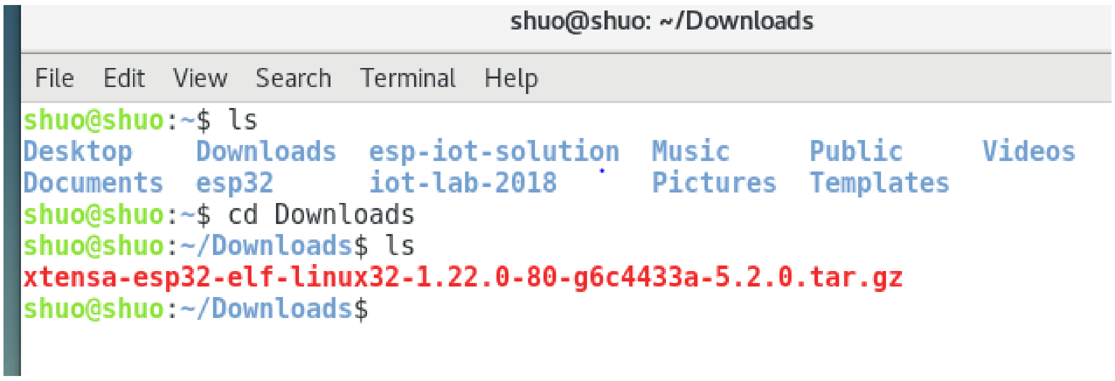
BME280: Final data sheet

Document revision	1.1
Document release date	May 07th, 2015
Document number	BST-BME280-DS001-10
Technical reference code(s)	0 273 141 185
Notes	Data in this document are subject to change without notice. Product photos and pictures are for illustration purposes only and may differ from the real product's appearance.

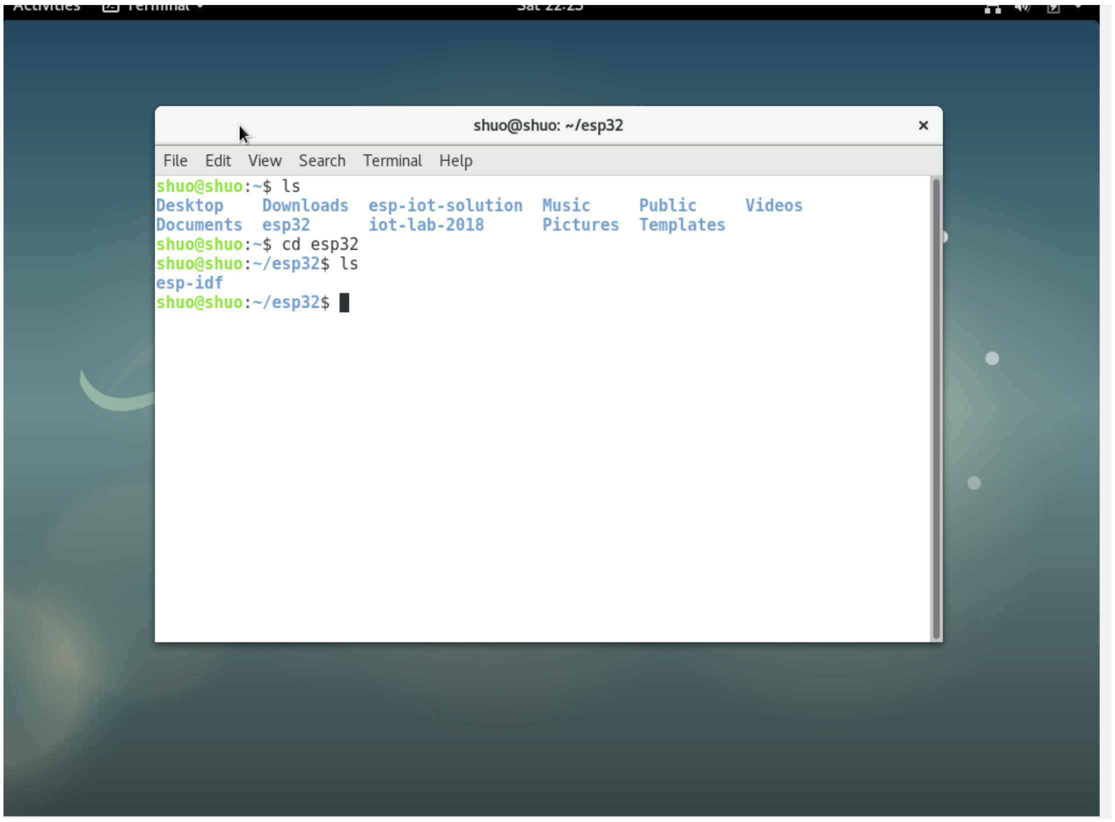
The BME280 is as combined digital humidity, pressure and temperature sensor based on proven sensing principles. The sensor module is housed in an extremely compact metal-lid LGA package with a footprint of only 2.5 × 2.5 mm² with a height of 0.93 mm. Its small dimensions and its low power consumption allow the implementation in battery driven devices such as handsets, GPS modules or watches. The BME280 is register and performance compatible to the Bosch Sensortec BMP280 digital pressure sensor (see chapter 5.2 for details).

The BME280 achieves high performance in all applications requiring humidity and pressure measurement. These emerging applications of home automation control, in-door navigation, health care as well as GPS refinement require a high accuracy and a low TCO at the same time.
The humidity sensor provides an extremely fast response time for fast context awareness applications and high overall accuracy over a wide temperature range.

We use Debian linux virtual machine to prepare the work place.



```
shuo@shuo:~$ ls
Desktop     Downloads  esp-iot-solution  Music     Public     Videos
Documents   esp32      iot-lab-2018      Pictures  Templates
shuo@shuo:~$ cd Downloads
shuo@shuo:~/Downloads$ ls
xtensa-esp32-elf-linux32-1.22.0-80-g6c4433a-5.2.0.tar.gz
shuo@shuo:~/Downloads$
```



```
shuo@shuo:~$ ls
Desktop     Downloads  esp-iot-solution  Music     Public     Videos
Documents   esp32      iot-lab-2018      Pictures  Templates
shuo@shuo:~$ cd esp32
shuo@shuo:~/esp32$ ls
esp-idf
shuo@shuo:~/esp32$
```

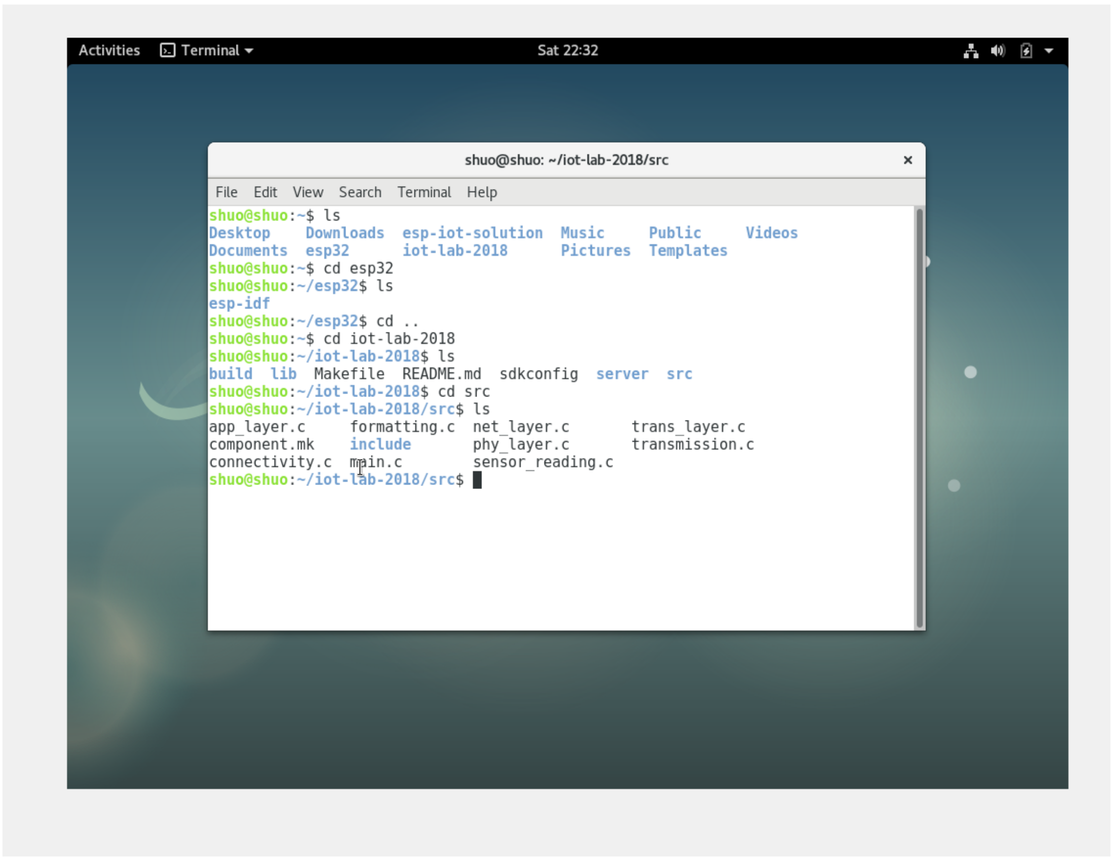
III. About the project:

An ESP32 will be responsible of collecting measures from one or optionally multiple sensors, then send them to a server, in the same local network.

Communication is based on the Light-weight Internet Protocol (LwIP) stack. This library is provided along with the ESP-IDF.

In this project ,we need to fill in the blanks in the files containing in the project.

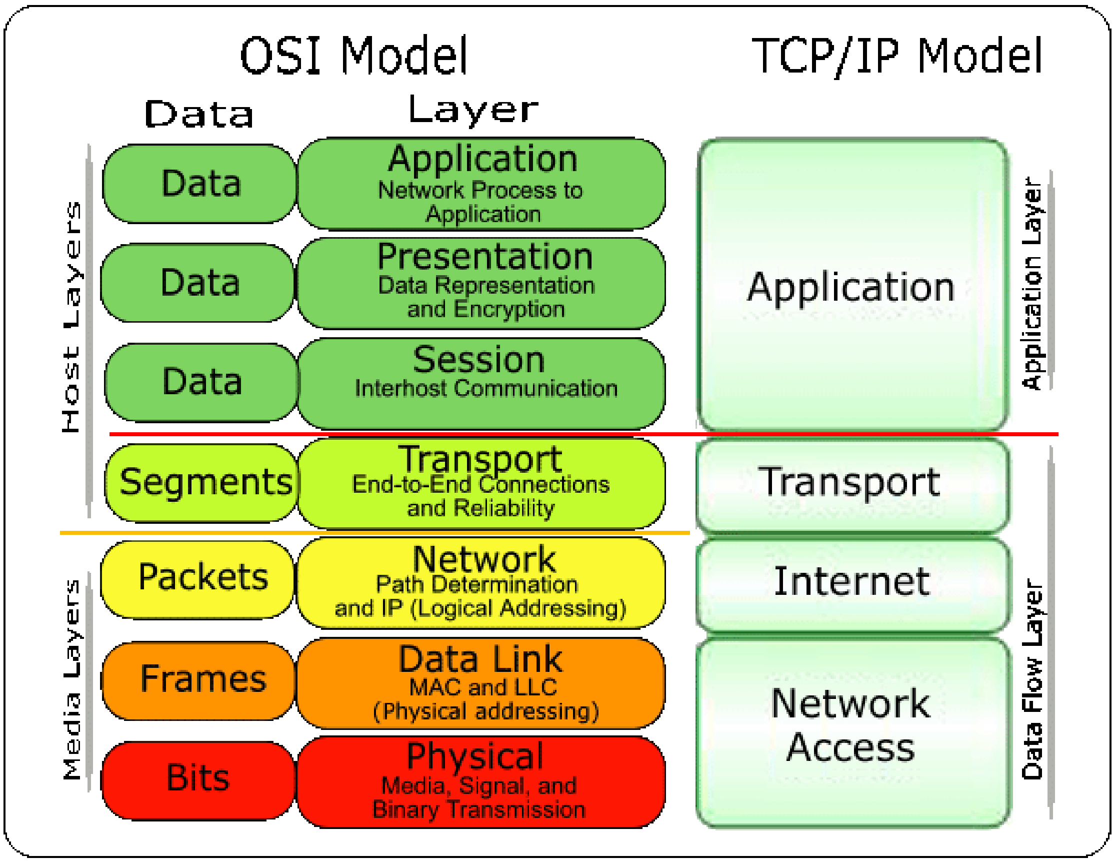
The .c files in the src document.


Activities
Terminal
Sat 22:32
shuo@shuo: ~/iot-lab-2018/src
File Edit View Search Terminal Help

```
shuo@shuo:~$ ls
Desktop    Downloads  esp-iot-solution  Music     Public     Videos
Documents  esp32      iot-lab-2018      Pictures  Templates
shuo@shuo:~$ cd esp32
shuo@shuo:~/esp32$ ls
esp-idf
shuo@shuo:~/esp32$ cd ..
shuo@shuo:~$ cd iot-lab-2018
shuo@shuo:~/iot-lab-2018$ ls
build  lib  Makefile  README.md  sdkconfig  server  src
shuo@shuo:~/iot-lab-2018$ cd src
shuo@shuo:~/iot-lab-2018/src$ ls
app_layer.c     formatting.c  net_layer.c       trans_layer.c
component.mk    include       phy_layer.c       transmission.c
connectivity.c  main.c        sensor_reading.c
shuo@shuo:~/iot-lab-2018/src$ 
```

First of all,we have filled in the phy_layer.c file,it is the physical layer of the Tcp/Ip protocol.After finished this part,the ESP32 can successful connected to my telephone hotpot.

The *physical layer* is the lowest layer of the OSI Reference Model; it is commonly abbreviated “PHY”. The physical layer is special compared to the other layers of the model, because it is the only one where data is physically moved across the network interface

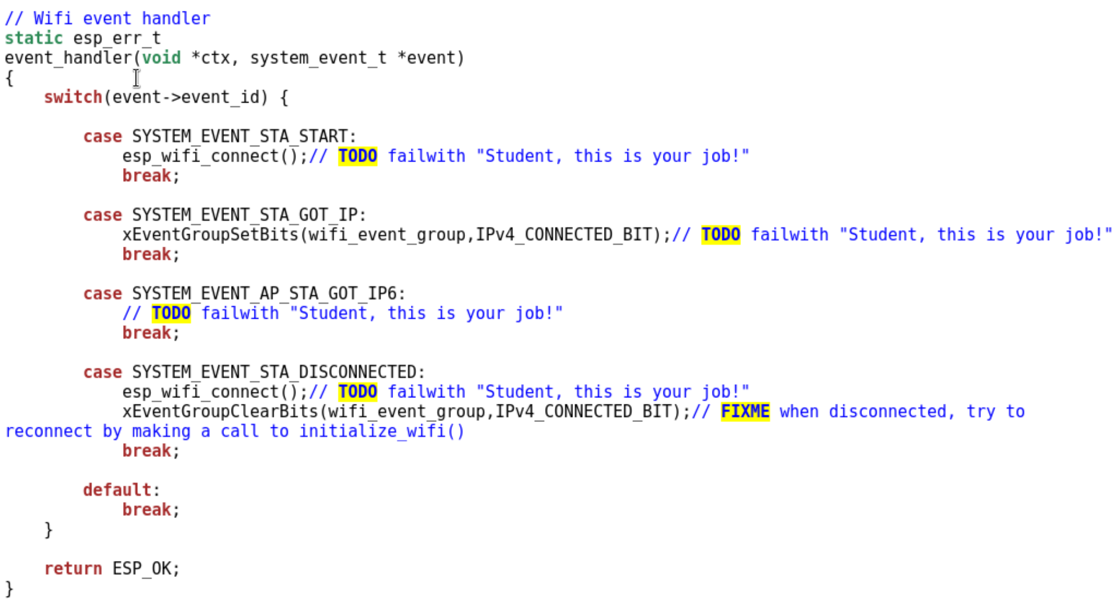


Graph of web protocol

All of the other layers perform useful functions to create messages to be sent, but they must all be transmitted down the protocol stack to the physical layer, where they are actually sent out over the network.

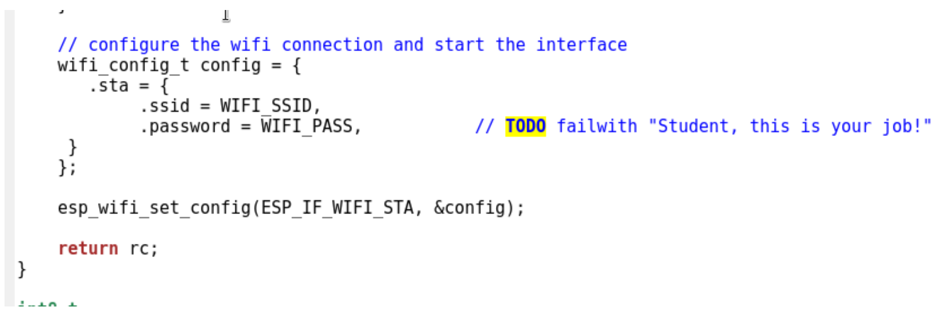
In this project,we need to initialize the phy_layer.c so that the development board (ESP32) can access to the hotpot,and then we can continue our projet. The physical layer is the fundamental of this internet protocol.

And we need to use the command “make flash” to ensure that all the programmation we have done is right.



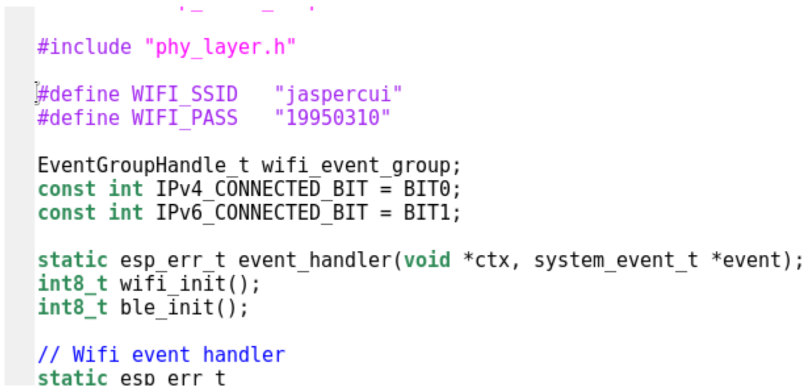
```
// Wifi event handler
static esp_err_t
event_handler(void *ctx, system_event_t *event)
{
    switch(event->event_id) {

        case SYSTEM_EVENT_STA_START:
            esp_wifi_connect();// TODO failwith "Student, this is your job!"
            break;

        case SYSTEM_EVENT_STA_GOT_IP:
            xEventGroupSetBits(wifi_event_group,IPv4_CONNECTED_BIT);// TODO failwith "Student, this is your job!"
            break;

        case SYSTEM_EVENT_AP_STA_GOT_IP6:
            // TODO failwith "Student, this is your job!"
            break;

        case SYSTEM_EVENT_STA_DISCONNECTED:
            esp_wifi_connect();// TODO failwith "Student, this is your job!"
            xEventGroupClearBits(wifi_event_group,IPv4_CONNECTED_BIT);// FIXME when disconnected, try to
reconnect by making a call to initialize_wifi()
            break;

        default:
            break;
    }

    return ESP_OK;
}
```

This is the IPv4 Internet Protocol.



```
    // configure the wifi connection and start the interface
    wifi_config_t config = {
        .sta = {
            .ssid = WIFI_SSID,
            .password = WIFI_PASS,           // TODO failwith "Student, this is your job!"
     }
    };

    esp_wifi_set_config(ESP_IF_WIFI_STA, &config);

    return rc;
}
```

This is the wifi config,we need to connect to the hot pot that we defined at the beginning of this file.

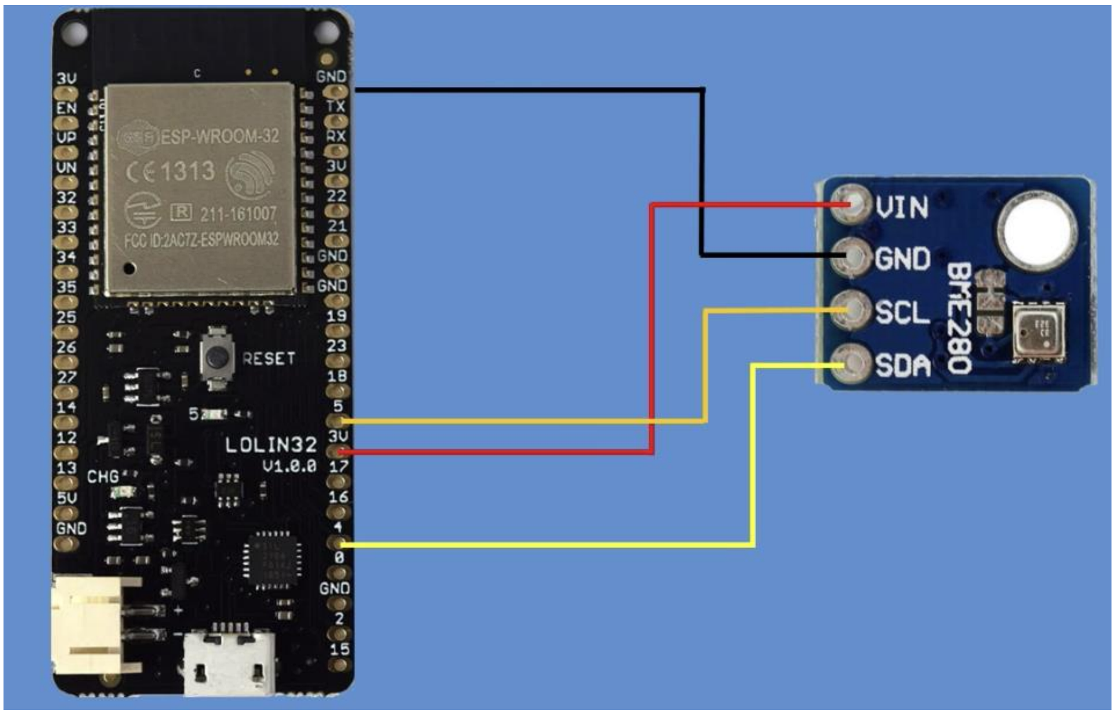


```
#include "phy_layer.h"

#define WIFI_SSID   "jaspercui"
#define WIFI_PASS   "19950310"

EventGroupHandle_t wifi_event_group;
const int IPv4_CONNECTED_BIT = BIT0;
const int IPv6_CONNECTED_BIT = BIT1;

static esp_err_t event_handler(void *ctx, system_event_t *event);
int8_t wifi_init();
int8_t ble_init();

// Wifi event handler
static esp_err_t
```

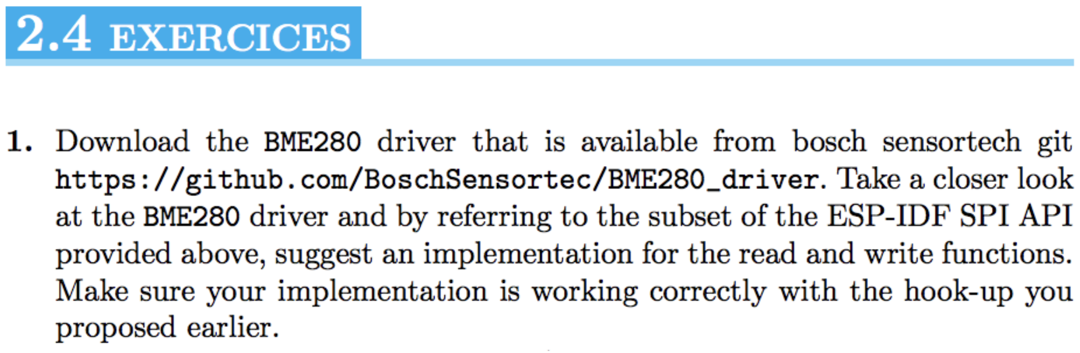
Here we define the wifi_ssid and wifi_pass.

Then we try to connect the BME280 with ESP32(it's still!!! Have some problem) :

We try the BME280 driver and try to connect the I2C device from the sensor_reading.c .



For Bme280 driver:
Before this lab we failed in this exercise



2.4 EXERCICES

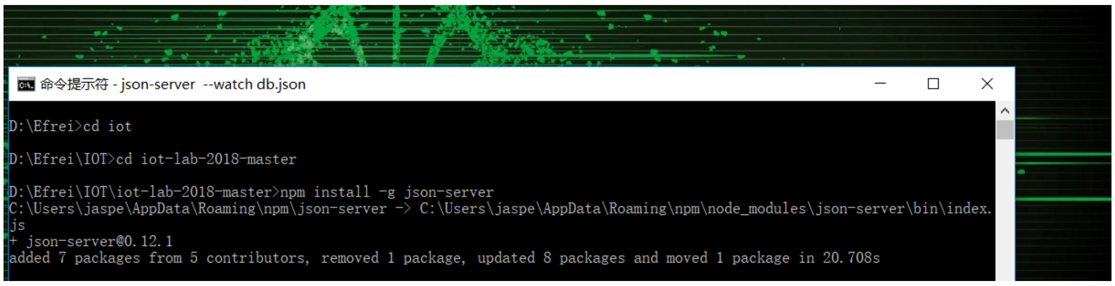
1. Download the BME280 driver that is available from bosch sensortech git `https://github.com/BoschSensortec/BME280_driver`. Take a closer look at the BME280 driver and by referring to the subset of the ESP-IDF SPI API provided above, suggest an implementation for the read and write functions. Make sure your implementation is working correctly with the hook-up you proposed earlier.

I think maybe it's why we didn't understand the code of BME280.
The report said: you can switch between SPI and I2C serial protocols ;
But when I want to fill the sensor.c I really don't how to do.

Now I think the problem maybe in the GPIO or SPI . We try to work in the Arduino.
First we download the Arduino IDE and then we configurate it. We follow the step of http://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-mac-and-linux-instructions/

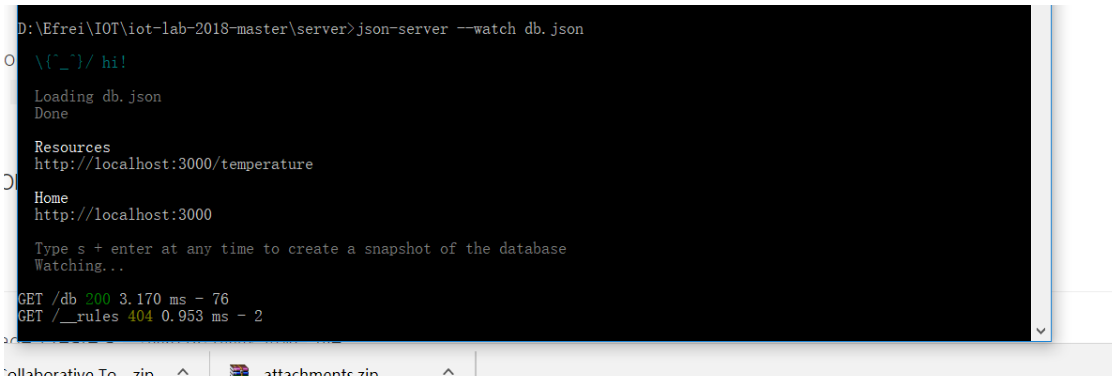
The problem is the version of python that we try to change the version python 2.7 but it does not work. So now we are trying to fix the problem two way but sadly we don't have a good result now.

JSON server:



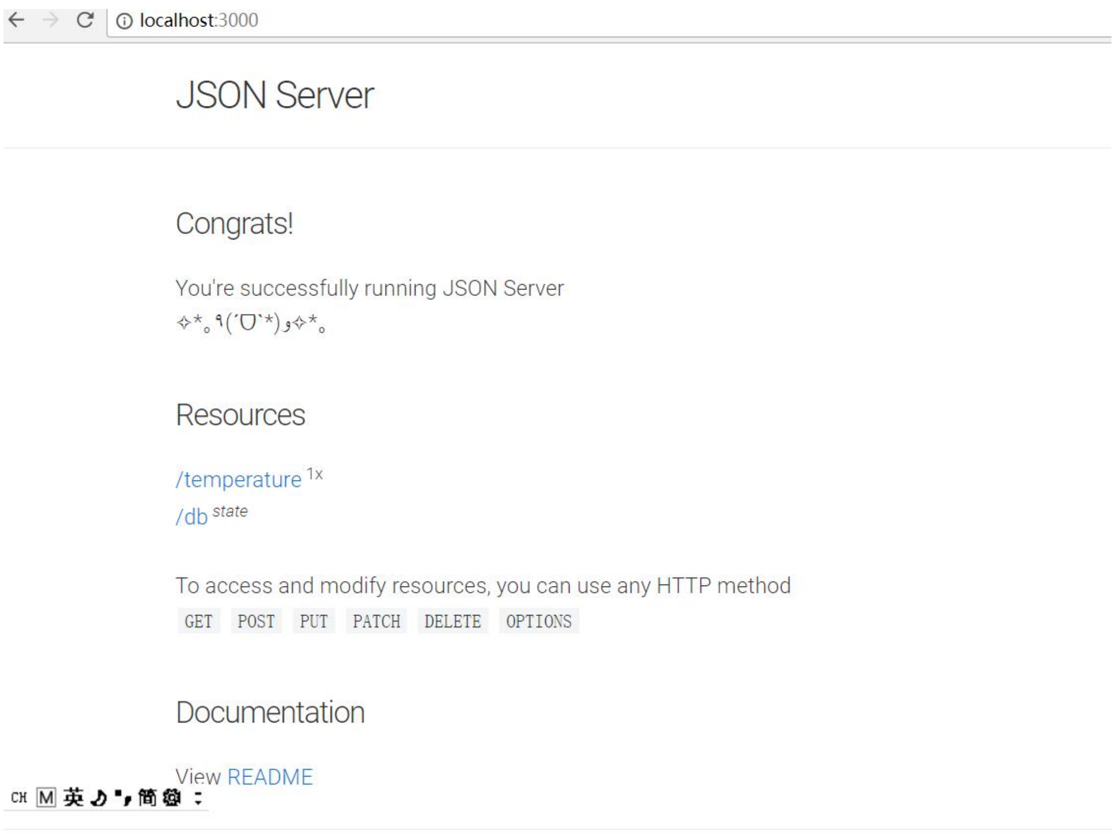
命令提示符 - json-server --watch db.json

```
D:\Efrei>cd iot

D:\Efrei\IOT>cd iot-lab-2018-master

D:\Efrei\IOT\iot-lab-2018-master>npm install -g json-server
C:\Users\jaspe\AppData\Roaming\npm\json-server -> C:\Users\jaspe\AppData\Roaming\npm\node_modules\json-server\bin\index.
js
+ json-server@0.12.1
added 7 packages from 5 contributors, removed 1 package, updated 8 packages and moved 1 package in 20.708s
```



```
D:\Efrei\IOT\iot-lab-2018-master\server>json-server --watch db.json

  \{^_^}/ hi!

  Loading db.json
  Done

  Resources
  http://localhost:3000/temperature

  Home
  http://localhost:3000

  Type s + enter at any time to create a snapshot of the database
  Watching...

GET /db 200 3.170 ms - 76
GET /__rules 404 0.953 ms - 2
```

Here we successfully run the Json server and toy database for the project



localhost:3000

JSON Server

Congrats!

You're successfully running JSON Server
✧*｡٩(ˊᗜˋ*)و✧*｡

Resources

/temperature 1x
/db state

To access and modify resources, you can use any HTTP method
GET POST PUT PATCH DELETE OPTIONS

Documentation

View README

Ⅳ.Conclusion

In this class,we have learned some basic knowledge of Internet Of Things,about how does it works.Besides, we do the TP of ESP32,it is a kind of development board of IOT.It’s the first time for us to prepare the workspace in the linux operation system with virtual machine.Thanks to this lesson,I recalled some basic operation of Linux.Moreover,I learned how to compile the whole project with ESP32 and toolchain..

However,we also meet some problems.During the time I prepared the workspace,it was really difficult because it has been a long time I didn’t use linux,I’m not familiar with that. Thanks to the help of teacher,we prepared the workspace.

After that,there were also the problem of compiling.In fact,we used very few time of language C after we finished the lesson language C in China ,and we are not familiar with that.But after we read some guideline in the internet,we solve some problems and fill in some blanks existing in the project.

Besides, there is also the problem of the electronic components,we have nearly 30 students in the class but we just have 3-5 BME280 sensors.Actually,we didn’t use the sensor until the last lesson of IOT.Although teacher explained the steps of the project,we still need enough time to understand and try it with the components.But after the last lesson,we can’t find where are the sensors.I asked some classmates in the class,but I can’t got the answer.I think that might be the reason why we haven’t finished the project yet.But we will continue our project after the deadline.And we will send you a better work after a few days.