**Individual Progress Report of**

**Low-cost Sensor for Seamless Road Quality Monitoring**

ECE 445 Design Document — Spring 2022

Individual Report by Yuhang Chen

Team Member: Yuhang Chen, Zhengwei Dai, Yichen Li, Yihang Yang

Advisor: Simon Hu

1. **Introduction**

**1.1 Team Project Overview**

The monitoring and maintenance of road surface are essential to improve driving comfort, transport safety and maintain infrastructure integrity. Traditional road monitoring is carried out on a regular basis by specially designed instrument vehicles, which require a lot of time and money and can only cover a limited proportion of the road network. Therefore, our group determine to design a road anomalies detection system combined with hardware components and software design. An acceleration sensor, a GPS module, an independent power supply and a WIFI module are connected to a microcontroller unit (MCU) to serve as the hardware detection system. This system will fetch the data from testing on road and transfer the data to the cloud server through the WIFI module. The software design includes the data preprocessing on the data received from the cloud server and the filtering and segmentation algorithm implementation to do extracting on features of sensor data, which is also supposed to be displayed in real time on a smartphone.

**1.2 Individual Responsibilities and Role**

In this project, the major part I am responsible is to establish the cloud server and the build the connection between the cloud server and the hardware components. Specifically, the table below shows the main tasks for me to focus on.

|  |  |  |
| --- | --- | --- |
| IDX | Task | Aspect |
| 1 | Physical Circuit Connection Design between the WIFI module and MCU | Hardware |
| 2 | Programming on the hardware driver and uploading codes to the MCU | Hardware-software Interaction |
| 3 | Building the front end and back end of the cloud server | Software |

Table 1. My Main Individual Responsibilities in this Project

1. **Individual Design Work**

**2.1 Design Consideration**

The design in Task 1 & 2 have changed a lot compared to the previous design since the Design Review. In the Design Review, our group considered to use a ESP8266 chip as the WIFI module and a stem32C103F chip as the MCU. We have gotten the ESP8266 and stem32C103F earlier. But we found that a TTL to USB converter was still needed to allow the stem32C103F to be connected with the computer, since the stem32C103F we bought only transmits TTL signal and we need to convert TTL signal to USB signal. So we decide to buy a CP2102 chip from TaoBao as the signal converter. However, due to the epidemic, we fail to get this chip. The university lab in our campus could not support any TTL to USB converter either while it could offer an integrated breadboard to solve this problem, which is called STEMTera BreadBoard. And we agree on using this breadboard after discussion. Basically, the STEMTera BreadBoard can be regarded as dual microcontrollers inserted in a breadboard while it provided with USB port and a port for external power supply. By using this board we solve the problem to connect the MCU with the computer. The details of the whole tolerance analysis according to this breadboard may be talked by my partners who are responsible for the sensor. But this breadboard could directly offer 3.3Vfor the WIFI module, so the problem of power supply for esp8266 has been basically solved.

As mentioned before, I am responsible for building the connection between the cloud server and hardware components while the use of STEMTera Breadboard make some change on the method that I drive the WIFI module. The tables below show the similarities and differences of the method that stem32 and STEMTera drive esp8266.

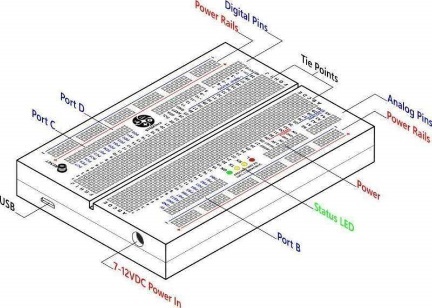
|  |  |  |
| --- | --- | --- |
| IDX | Similarities | Discription |
| 1 | Programming Software | Both can support varieties of programing IDE (MDK-ARM, Arduino, IAR, etc.) In this project, I choose Arduino IDE. |
| 2 | Used Command | Both can use AT commands to set WIFI module on STA mode and initialize TCP connection. |

Table 2. The Similarities between Stem32C103F and STEMTera Breadboard

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IDX | Differences | Stem32C103F | | STEMTera BreadBoard | |
| 1 | Hardware Role | Stem32C103F only serve as the microcontroller for esp8266. | | STEMTera not only serve as the microcontroller for esp8266 but also provide power supply for esp8266. | |
| 2 | Pin Connection in the Circuit | Stem32C103F | Esp8266-01 | STEMTera | Esp8266-01 |
| Out of stem32 | VCC | 3.3V | VCC |
| GND | GND | GND | GND |
| PA3 | TX | Pin 8 | TX |
| PA2 | RX | Pin 9 | RX |
| 3 | Programming Library | ESP8266WiFi.h by stem32  [1] | | esp8266.h by esp8266 community | |

Table 3. The Differences between Stem32C103F and STEMTera Breadboard

**2.2 Physical Diagram**



Esp8266

TTL Connection

USB Connection



Cloud Server

Laptop Backend

MCU

WIFI Client

Data Transmission

Figure 1. Physical Diagram on the Building the Connection

between the WIFI Module and the Cloud Server

Step 1. Build physical hardware connection

LED On, WIFI module is recognized by computer

LED Off

Step 2. Initialize ESP8266 Model

AT command Test successfully in COM

AT command Failed

Step 3. Establish TCP/IP and use python socket to build TCP/IP

Fail to send data to the IP Address

Succeed to send data to the IP Address

Step 4. Use python flask and html to build the cloud server

Step 5. Final Test

Show data in the server

Fail to Show data

Figure 2. Flow Chart on the work on the WIFI Module and the Cloud Server

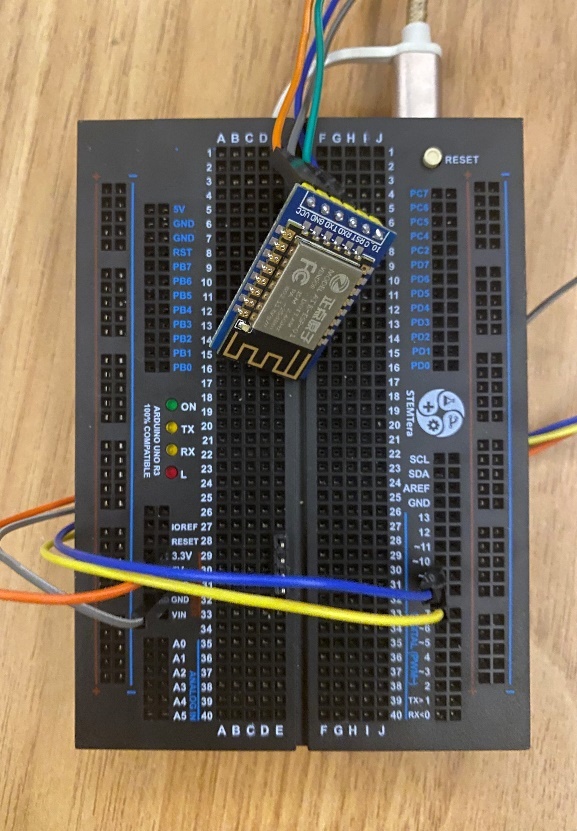
**2.3 Test & Verification**

**2.3.1** **Test Plans and Procedures**

|  |  |
| --- | --- |
| Date | Procedures |
| 3/14-3/20 | Collect all of the hardware components in the project from Taobao and the university lab |
| 3/21-3/27 | Build the physical connection and weld the used pin ports in the WIFI module esp8266; Test on the connection between WIFI module and the computer backend |
| 3/28-4/3 | Test on the Initialization on the WIFI module, set WIFI module to STA Mode and Initialize TCP link by AT command inserted in esp8266 |
| 4/4-4/10 | Test on establishing TCP/UDP to set IP and use python socket to build TCP/UDP |

Table 4. Test Plans and Procedures of My Individual Work Process

**2.3.2 Modular Testing**

****

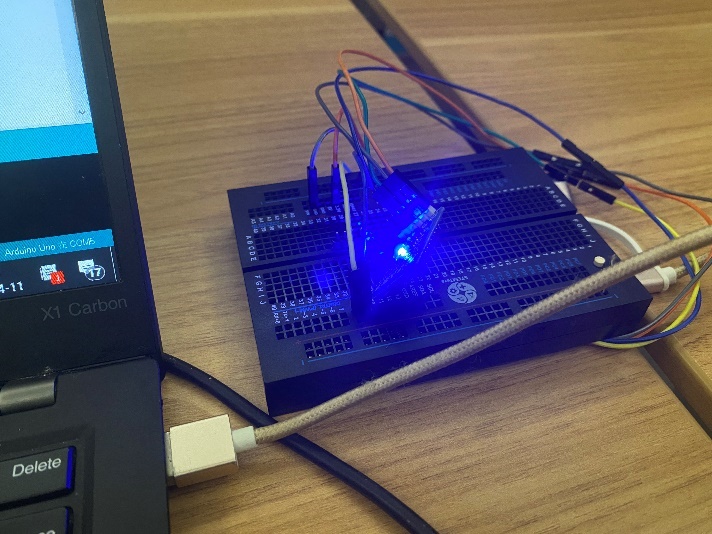


Figure 4. Connection between the hardware components and the laptop

Figure 3. Circuit Connection of esp8266 and STEMTera BreadBoard

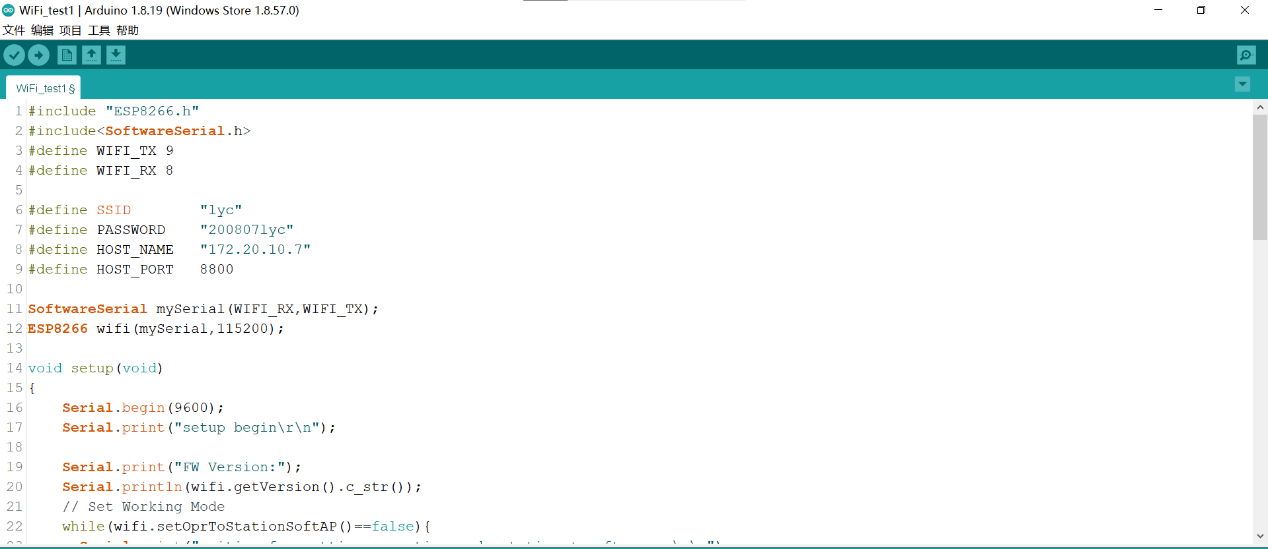


Figure 5. Code Snips of WIFI modular Test

**2.3.3** **Test Results**

|  |  |  |  |
| --- | --- | --- | --- |
| IDX | Test Content | Result Status | Failure Analysis |
| 1 | Connect Pin 0 and Pin 1 on STEMTera Breadboard with RX and TX port on esp8266 and test on whether the connection between WIFI module and the MCU is successful by using serial port print in Arduino IDE | Fail | By learning that the pin 0 (RX) is the receiver part of this communication system for the serial port monitor in Arduino IDE, I got that we can not connect pin 0 with the slave(WIFI module or other chips) and require the serial port monitor to work. Thus we may choose other pins in STEMTera Breadboard for connection. [2] |
| 2 | Connect Pin 8 and Pin 9 on STEMTera Breadboard with RX and TX port on esp8266 and test on whether the connection between WIFI module and the MCU is successful by using serial port print in Arduino IDE | Success | NULL |
| 3 | Use AT command to initialize WIFI module as AP mode to send data to the cloud server | Fail | Learning that we cannot set the WIFI module as AP mode to make it serve as the server, instead, we should set it to STA mode to make it as the client and use the client to send data to the server. [3] |
| 4 | Use AT command to initialize WIFI module as STA mode to send data to the cloud server | Success | NULL |
| 5 | Use socket in python to establish TCP link to show the received data in the server (in Wins 10 system) | Fail | There exist some Network Protection Protocol in Wins 10 System to block the links. [4] |
| 6 | Use socket in python to establish TCP link to show the received data in the server (in linux system build in urbuntu) | Success | NULL |

Table 5. Test Trials and Results during the process of

Connection between WIFI module and the Cloud Server

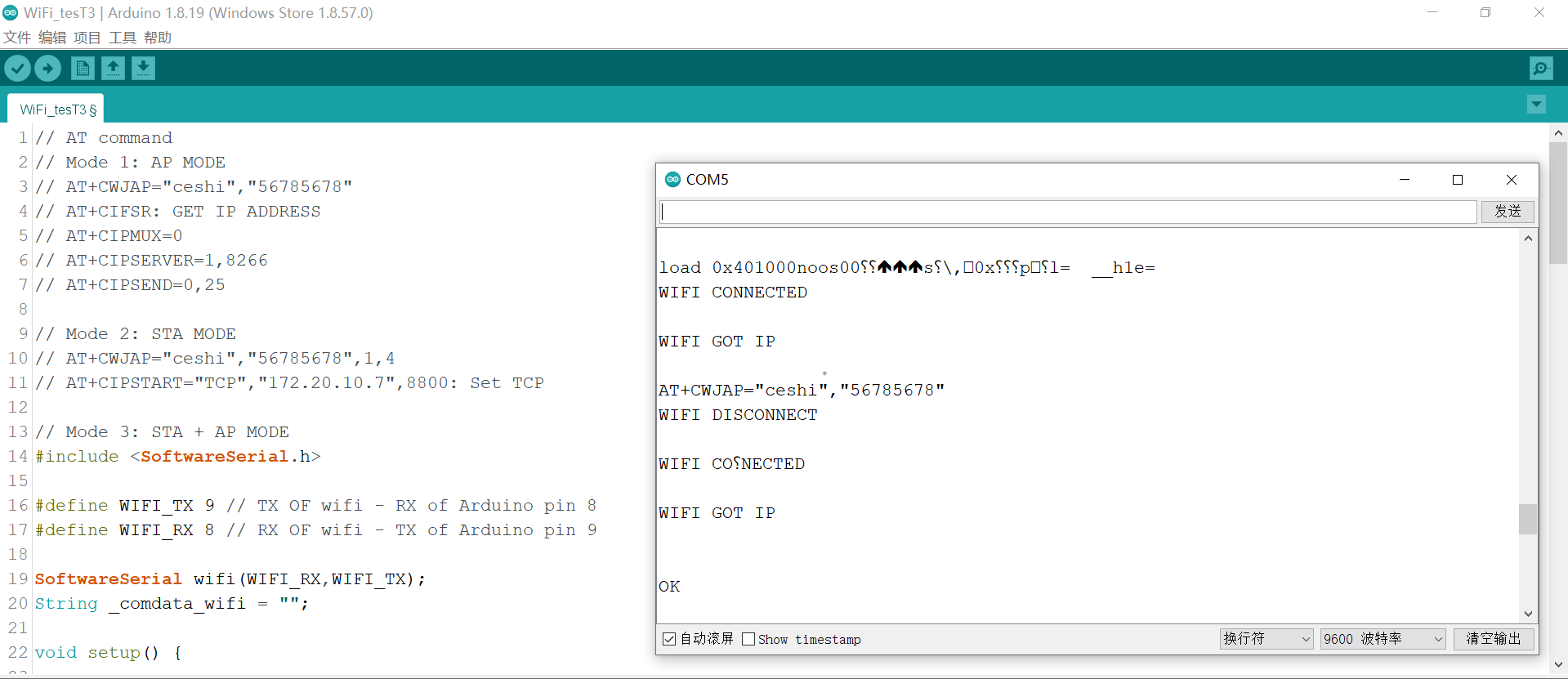


Figure 6. Code Snips and Serial Port Monitor Show of WIFI STA Mode Test

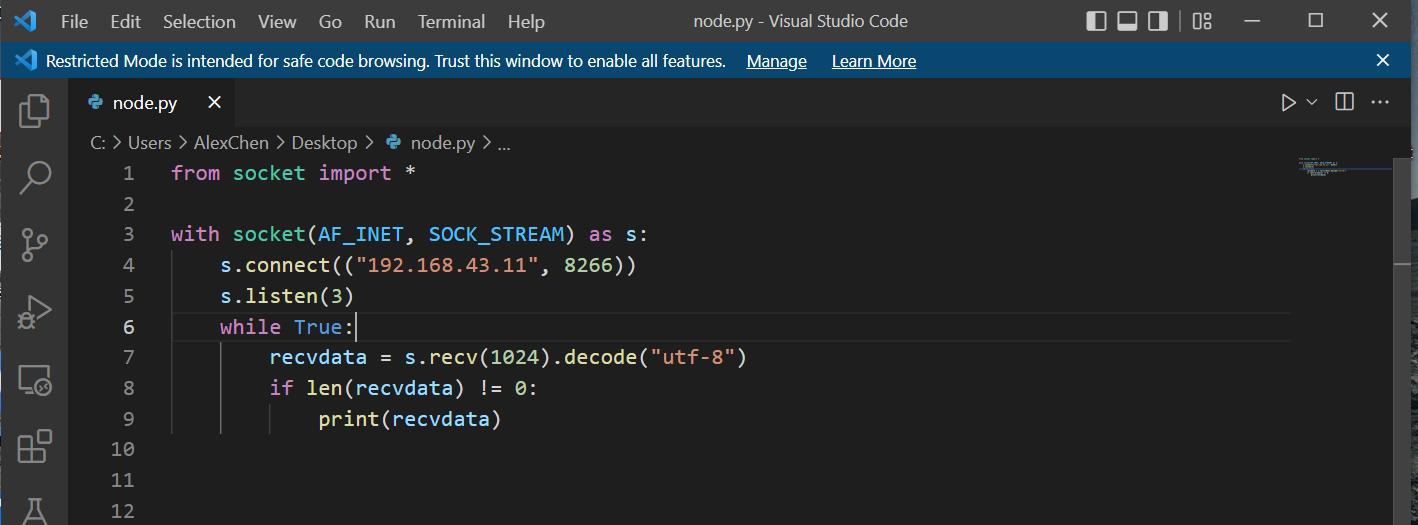


Figure 7. Code Snips of Using Socket to Show the Received Data in Cloud Server

**3.Conclusion**

**3.1 Self-assessment**

My Individual workload is enough for me to contribute myself on this project. Since our group project has both hardware design and software design, meanwhile with hard-soft interaction implementation work, our total workload is very large. There is a clear division of work in our group. So far I am satisfied with the division and the progress of our project. We discussed every week in discussion room and work together if some team members have some problem in some part of the project work like the sensor testing, etc.

I think I am working on schedule, at least not behind the schedule because I have succeeded to make the hardware and software work together and I can focus more on the software programming about how to build the cloud server efficiently. Although there is still a lot of work to do, I believe my part of work could be done according to the schedule plan and I may even have remaining time to help my partners on the filtering and segmentation algorithm.

**3.2** **Plan for Remaining Work**

|  |  |
| --- | --- |
| Date | Procedures |
| 4/11-4/24 | Succeed to establish the whole TCP link and transmit sensor data from the MCU to the cloud server |
| 4/25-5/8 | Write html page and use flask in python to establish the front end and back end of the cloud server |
| 5/9-5/22 | Final Test and make adjustment on the cloud server and help on data processing |

Table 6. Plan for Remaining Work from Apri.11

**4.Reference**

[1] Connecting ESP8266 with STM32F103C8: Creating a Webserver. Pramoth Thangavel. Oct 25, 2018 <https://circuitdigest.com/microcontroller-projects/interfacing-esp8266-with-stm32f103c8-stm32-to-create-a-webserver>

[2] Troubles in Connecting esp8266-01 with Arduino in forum Arduino.cc <https://forum.arduino.cc/t/troubles-connecting-esp8266-01-with-arduino-mega-2560/646624>

[3] Esp8266, Use Python to Establish Client Connections and Realize Data Mutual Transmission (Transparent Transmission). Tian, Xiaohua. Feb 16, 2020. <https://blog.csdn.net/qq_44179528/article/details/104346620>

[4] Fail to Connect esp8266 ip address by CSDN blog. Douran. May 28, 2019. <https://blog.csdn.net/weixin_44986800/article/details/90637668?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1.pc_relevant_default&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1.pc_relevant_default&utm_relevant_index=2>