COEN 315 Final Project Report

Kai Chieh Liu
Sante Clara University
Santa Clara, United State
kliu2@scu.edu

Ting-Chi Yeh
Sante Clara University
Santa Clara, United State
tyeh@scu.edu

Wen-Han Chang
Sante Clara University
Santa Clara, United State
wchang1@scu.edu

***Abstract*—This report presents the detail of design and implement of our team project for COEN 315: Web Architecture & Protocols. Since COEN 315 is a course that teaches methodology of IoT related technologies, we want to use this opportunity to learn how to develop interactive web application using IoT technologies. The architecture of this project includes a web application, RESTful server, ESP chips, camera, motion sensor, motor smart robot car.**

Keywords—coen315t, IoT, RESTful API , ESP8266, ESP32

I. INTRODUCTION

A. Background

Based on our research, the commercial security cameras on the market have to be installed on wall and have limitation on the vision range[15]. To monitor the entire house, the customer has to purchase multiple cameras and installs them all around the house. It becomes a waste of resources when the scale of environment area gets larger. The solution our team provide for this problem is to have a security cameras which can move around the house—CarPet.

B. Objective and Purpose

The objective of our team is to develop a remote controllable house security guard, which can help people ensure the security of their house when no one at home. By applying controllable mobility, the security guard minimize its blind angle and increase its flexibility.

II. DESIGN

A. Structure

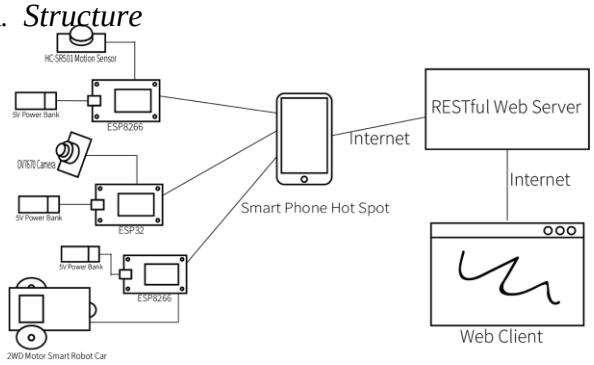

Figure 1 Structure Diagram of CarPet

The general structure of our project, CarPet is showen in figure 1. The skeleton of CarPet is builded on ESP8266 chip[3]. The ESP8266 serves as a bridge between the hardware components and the web server. To generate the full functionality of the automatic house security guard, we construct a motion sensor and a camera with a Motor Smart Robot Car. The motor robot car provides the mobility of the camera. These devices are connected to the web server. We designed and implemented web server by Node.js[2] and run it on an Amazon EC2 instance. The web application is written in HTML and communicates to the server using JavaScript[1].

Figure 2 ESP32

Figure 3 ESP8266

These ESP chips are hearts of CarPet. They provide other hardware components capability of connecting to internet. The model of ESP8266 we using is ESP-12E NodeMCU. We have two ESP8266 in a CarPet. One is connected to a car. It helps making the car becomes remotely controllable. Another is connected to motion sensor to making it able to send data to our server for the use of notification. During research, we found it needs more bandwidth to implement live video streaming. Hence, we replaced ESP8266 with ESP32[4], a successor to ESP8266, and connected it to the camera. Figure 2 and figure 3 are showing the picture of ESP32 and ESP8266 chips.

B. *Implement Challenge*

1) *Slow Video*

The camera model is OV7670 without FIFO[9, 10, 11, 12]. We connected it to internet via ESP32. The quality of live video was horrible. It is only a frame per 4~5 seconds. The reason could be 1. Taking picture is time consuming. 2. Sending a image needs long time. 3 Both. Our experiment shows taking picture only spends about 70 milliseconds. Obviously, the problem is sending picture.

To improve quality of video streaming, we tried following methods:

- ESP32: Since the problem may caused by limitation of wifi speed. We replaced ESP8266 with ESP32 to get faster transmit speed. Unfortunately, the problem is still there.
- UDP socket: It has better transmission speed performance than using HttpClient library. However, we couldn't assemble data packet received on the server back to image due to loss of data during transmission.
- TCP socket: It has better transmission speed performance than using HttpClient library. However, the content or header of picture assemble on the server side is corrupted occasionally. Hence, this is not a viable solution.

2) *Motion Sensor voltage Problem*

The motion sensor is originally designed for powering with 5 Voltage. However, we design to supply the power of the motion sensor with 3.3 voltage from a ESP8266 chip. Powering it with 3.3 voltage, we found that sometimes when the output signal should be low value, we got high value. The problem is that the output signal lies in undefined region[5], shown in figure 4. The solution is adding a voltage divider[6](figure 5) to make the low value output below VIL.

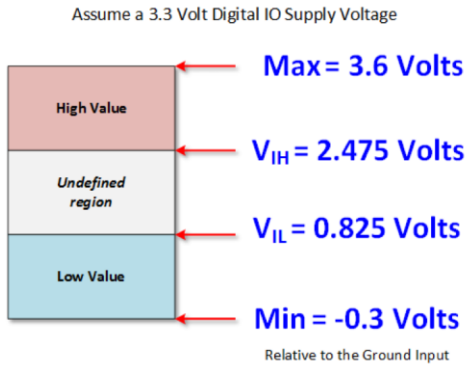

Figure 4 Voltage Levels

The output signal voltage of the motion sensor is divided by two resistors. The first one is 1kΩ, and the other one is 2.2kΩ. To drop the output voltage out of the undefined region, we got the output signal from the end of the first one resistor.

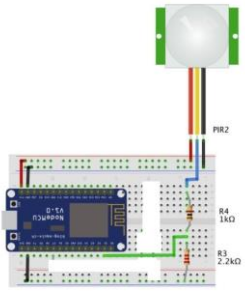

Figure 5 Voltage Divider

III. Desigh Methodology and Hardward Setup

A. *Server*

In order to host CarPet website, we built our server in Apache2 and Node.js and run it on an Amazon EC2 server. Apache2 forwards http and https request to Node.js server. Node.js server handles GET request from clients and send corresponding file or data back to clients. It also allows our hardware components upload data, such as image from camera, through ESP chips and POST request. In addition, to notify client that a motion sensor detects an abnormal activity, Node.js server uses web socket to keep a connection with a client.

B. *Camera*

Table 1 shows pin connections between OV7670 and ESP32. Since we using OV7670 without FIFO, we can't implement live video streaming with video frame buffer memory. We achieved live video streaming by continuously taking a picture and sending it to our Node.js server. The way we deliver image to the server was using HttpClient library and its POST method. To using POST method provided by HttpClient library, we first established connection to our server by calling "begin" method. Second, we set content-type in the header with "text/plain" by calling "addHeader" method. Then we called "POST" method to send an image to the server. The reason of using text/plain is that our server only stores image data and serve it to clients when receiving request. The server doesn't have to know what data it receives and sends is.

Table 1 Pin Connections between OV7670 and ESP32

OV7670 Pin	Description	ESP32 Pin
VCC	3.3v Power supply	3.3V
GND	Power Ground	GND
SCL	Two-Wire Serial Interface Clock	GPIO 22
SDA	Two-Wire Serial Interface Data I/O	GPIO 23
VSYNC	Active High: Frame Valid; indicates active frame	GPIO 21
HREF	Active High: Line/Data Valid; indicates active pixels	GPIO 19
PCLK	Pixel Clock output from sensor	GPIO 5
XCLK	Master Clock into Sensor	GPIO 18
D0	Pixel Data Output 0	GPIO 13
D1	Pixel Data Output 1	GPIO 12
D2	Pixel Data Output 2	GPIO 14
D3	Pixel Data Output 3	GPIO 27
D4	Pixel Data Output 4	GPIO 26
D5	Pixel Data Output 5	GPIO 25
D6	Pixel Data Output 6	GPIO 33
D7	Pixel Data Output 7	GPIO 32

C. *Motion Sensor*

Figure 6 HC-SR501

We adopted HC-SR50 (figure 6)[6] as our motion sensor. The motion sensor is adjustable for sensor range and time delay. The pins description of HC-SR501 is labeled in figure 7. The motion sensor has one output pin for transmitting detection signal, and it is linked to a ESP8266 chip. Every time the sensor is triggered, a message will send to server to notify Carpet user that something is detected by the motion sensor. Table 2 presents the pins connections between Motion sensor and ESP8266.

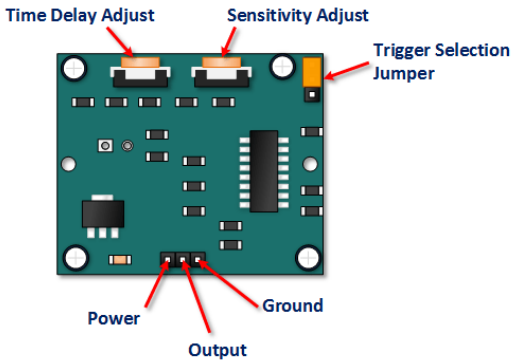

Figure 7 HC-SR501 Pins

Table 2. Pin Connections between Motion sensor and ESP8266

Motion sensor pins	Description	ESP8266 Pins
Power	5 VDC supply input	3.3V
Output	Low when no motion is detected. High when motion is detected.	GPIO 5
Ground	Ground input	GND

D. *Robot Car*

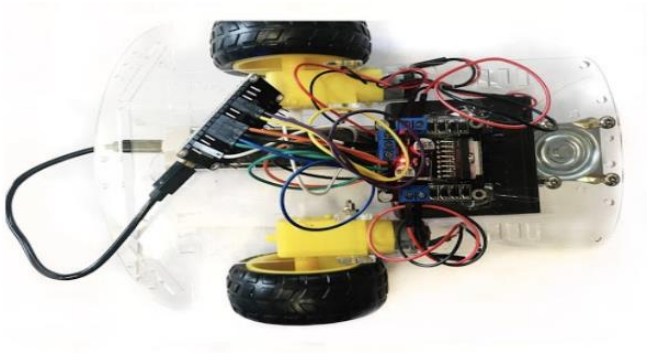

Figure 8. Robot car

The robot car (figure 8)[13] is composed of five main parts - a ESP8266 chip, a motor controller L298N H-bridge (figure 9)[14], a car chassis, two motors, and three wheels. The ESP8266 chip connects to the motor controller L298N H-bridge, and the motor controller connects to two motors. When the ESP8266 chip receives a car move command from the server, it delivers move signals to motor controller L298N H-bridge to drive robot cars. Table 3 and table 4 are showing the connections between L298N H-bridge and Parts and the connections between L298N H-bridge and ESP8266.

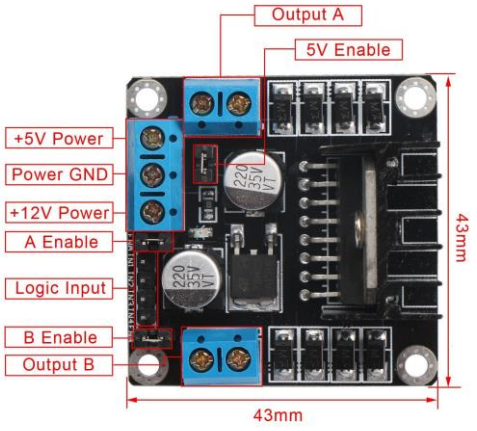

Figure 9. Motor controller L298N H-bridge

Table 3. Connections between L298N H-bridge and Parts

L298N H-bridge	Description	Part
+5V	Power supply	Connect to battery
GND	Ground input	Battery ground
Output A	Control right motor	Connect to right motor
Output B	Control left motor	Connect to left motor

Table 4. Connections between L298N H-bridge and ESP8266

L298N H-bridge pin	Description	ESP8266 Pin
GND	Ground input	GND
Motor A enable	Motor A enabler	GPIO 14
Motor B enable	Motor B enabler	GPIO 12
IN 1	Logic input	GPIO 15
IN 2	Logic input	GPIO 13
IN 3	Logic input	GPIO 2
IN 4	Logic input	GPIO 0

IV. Result of Exploration

The result of our product can be separated to two major parts: application and physical product.

A. *Application*

Through the application, user can remotely control House Security Guard, monitor houses by watching live video and get notification if there are any unusual activities in the house. Our Web application is written in HTML, CSS, and JavaScript and is hosted on Amazon EC2 instance. The application contain a traditional web page (figure 10) and mobile hybrid web application(figure 11). We used PhoneGap to transform the web page into mobile application. A sleeping dog is waiting on the welcome page when the user first lunch to the application(figure 10a and 11a). Clicking the picture, the dog will wake up

indicated the security guard-Carpet is ready to take the user requests. On this page, if there is any life movement detected by the motion sensor on the car, Carpet will bark to alert the user(figure 10b and 11b).

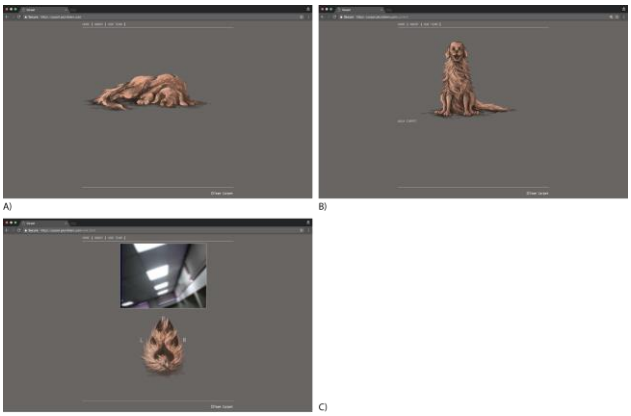

Figure 10 Web Application (Web Page)

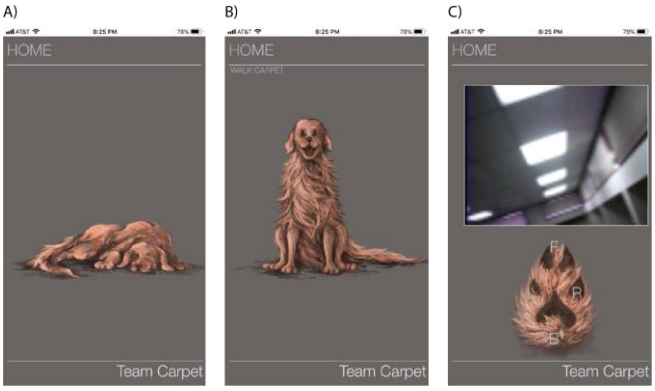

Figure 11. Web Application (Mobile APP)

After the user is call for action by Carpet, the user can follow the button on this page to view the live streaming took by the camera. On the screaming page, users can use the navigation panel below the video window to drive the direction of the car (figure 10c and 11c).

B. Phical Product

The photo of our end final Automatic House Security Guard is shown in figure 12. It use 2WD Motor Smart Robot Car as source of movement ability. A build-in camera provide live video for house owner to know if there are something abnormal. On the roof of the car, there is a motion sensor which detects unusual activities around it so the House Security Guard can send notification to Web application.

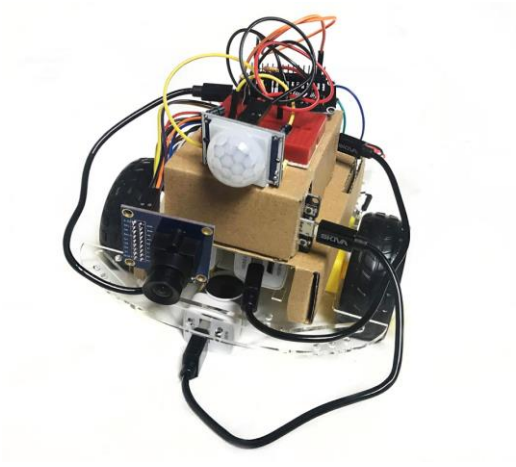

Figure 12 Final Prototype of CarPet

V. CONCLUSION

Designing and Implementing Carpet from scratch, we dig deeply on exploring how the web-related technologies (IoT, Cloud, Front-end, Back-end, DNS …etc.) this day can influence people's life from a more technical and business perspective. We are able to address the answers for the question we had before this project about how people can benefit from this IoT product. We used easy-to-purpose low-cost materials and made a simple home product which connect to the web services and serves multiple functions. The average price for a commercial security camera is 100 to 500 dollars [15]. The total cost of our product is under 50 dollars. ESP8266 is a simple and cheap chip but with great functionality. We learned how to integrate it with other hardware devices in this project. Also, we learned how to design and implement RESTful API and applications. This group-based project encourages us to discuss our opinions on the different technology and do research on the latest web-of-things technologies.

VI. RETROSPECTIVE AND FUTURE WORK

There are still numerous of update we can do to improve our project. Currently, it can only alert user by showing message and sound notification on the web page. The user has to be at the particular page to see the notification. It will be more practical if we can change the notification technique, so the user does not have to opening the application to see the notification. From the demo, no one can omit the quality of video we have for our life screaming function right now. It is a major issue for our project since live video is one of our target function. Our product currently provides only one car for users. To provide multiple cars for users, we will integrate registration system and database into our product.

ACKNOWLEDGMENT

We thank professor Amr Elkady for assistance with methodology on web and internet architecture and technique on ESP chips. Overall, we greatly appreciated obtaining this opportunity to exercise what we learn from class because practice makes perfect.

REFERENCES

[1] "W3Schools Online Web Tutorials", W3schools.com, 2018. [Online]. Available: https://www.w3schools.com/. [Accessed: 24- Mar- 2018].

[2] N. Foundation, "Node.js", Node.js, 2018. [Online]. Available: https://nodejs.org/en/. [Accessed: 24- Mar- 2018].

[3] "esp8266/Arduino", GitHub, 2018. [Online]. Available: https://github.com/esp8266/Arduino. [Accessed: 24- Mar- 2018].

[4]]"Espressif Systems", GitHub, 2018. [Online]. Available: https://github.com/espressif. [Accessed: 24- Mar- 2018].

[5] "IoT Motion Detector With NodeMCU and BLYNK", Instructables.com, 2018. [Online]. Available: http://www.instructables.com/id/IoT-Motion-Detector-With-NodeMCU-and-BLYNK/. [Accessed: 24- Mar- 2018].

[6] "Arduino HC-SR501 Motion Sensor Tutorial", Henry's Bench, 2018. [Online]. Available: http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-hc-sr501-motion-sensor-tutorial/. [Accessed: 24- Mar- 2018].

[7] "ESP8266EX GPIO High and Low Input Thresholds", Henry's Bench, 2018. [Online]. Available: http://henrysbench.capnfatz.com/henrys-bench/arduino-projects-tips-and-more/esp8266ex-gpio-high-and-low-input-thresholds/. [Accessed: 24- Mar- 2018].

[8] "ESP8266 - NURDspace", Nurdspace.nl, 2018. [Online]. Available: https://nurdspace.nl/ESP8266. [Accessed: 24- Mar- 2018].

[9] "igrr/esp32-cam-demo", GitHub, 2018. [Online]. Available: https://github.com/igrr/esp32-cam-demo. [Accessed: 24- Mar- 2018].

[10] "ESP32 I2S Camera (OV7670) – bitluni's lab", Bitluni.net, 2018. [Online]. Available: http://bitluni.net/esp32-i2s-camera-ov7670/. [Accessed: 24- Mar- 2018].

[11] Play-zone.ch, 2018. [Online]. Available: https://www.play-zone.ch/en/fileuploader/download/download/?d=1&file=custom%2Fupload%2FFile-1402681702.pdf. [Accessed: 24- Mar- 2018].

[12] Haoyuelectronics.com, 2018. [Online]. Available: http://www.haoyuelectronics.com/Attachment/OV7670%20+%20AL422B%28FIFO%29%20Camera%20Module%28V2.0%29/OV7670%20Implementation%20Guide%20%28V1.0%29.pdf. [Accessed: 24- Mar- 2018].

[13]]"NodeMCU ESP8266 WiFi Robot Car Controlled by Application (Wi...", Mertarduinotutorial.blogspot.com.tr, 2018. [Online]. Available: http://mertarduinotutorial.blogspot.com.tr/2017/05/nodemcu-esp8266-wifi-robot-car.html. [Accessed: 24- Mar- 2018].

[14] Amazon.com, 2018. [Online]. Available: https://www.amazon.com/gp/product/B00CAG6GX2/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1. [Accessed: 24- Mar- 2018].

[15] "Amazon.com: Remote Home Monitoring Systems: Electronics", Amazon.com, 2018. Online]. Available: https://www.amazon.com/s/ref=s9_acsd_dnav_hd_bw_b2CLo_ct_x_ct02_w?node=7161093011&pf_rd_m=ATVPDKIKX0DER&pf_rd_s=merchandised-search-11&pf_rd_r=1R3KRYVGRXM5V98J0AHD&pf_rd_t=101&pf_rd_p=490b5b0e-1aa1-5ead-ba61-77545d54848f&pf_rd_i=524136 . [Accessed: 24- Mar- 2018].