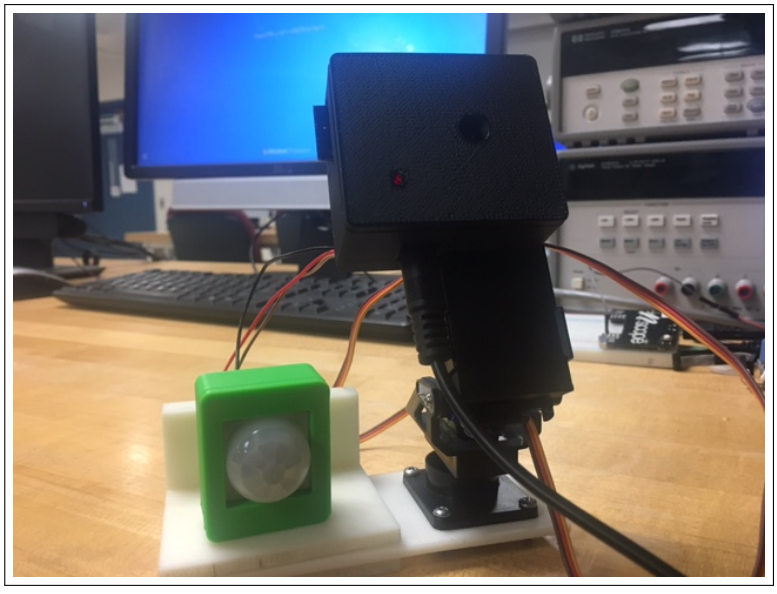
NORTHWESTERN UNIVERSITY

EECS 495 ENGINEERING SYSTEM DESIGN FINAL PAPER

Power House Surveillance

Author:
Hannah EMNETT
Karan SHAH

Instructor:
Dr. Ilya MIKHELSON



June 9, 2017

Northwestern University

Abstract

Electrical Engineering and Computer Science (EECS)

Power House Surveillance

by Hannah EMNETT
Karan SHAH

With burglaries on the rise, home security is of the utmost importance. Around 2,000,000 home burglaries are reported each year in the United States. Nearly 66 percent of all burglaries break-ins are through window or door. We present a novel and cost effective solution "Powerhouse Surveillance" consisting of a modified controllable webcam and motion detection system. The product demonstrates threefold functionality. The webcam can detect motion through a PIR sensor to start displaying images. The webcam, mounted on a pan-tilt base, can be manually controlled from a website whose server is hosted on a Raspberry Pi. Finally, the webcam can analyze the incoming images and recognize an object as the object is moving in the images. By implementing such a system, the homeowner can view who has entered his home or if there a potential burglary taking place. We describe the different components and the design process in detail in the report. We presented our system at the final exam fair along with the poster. The poster is shown in Figure 1.

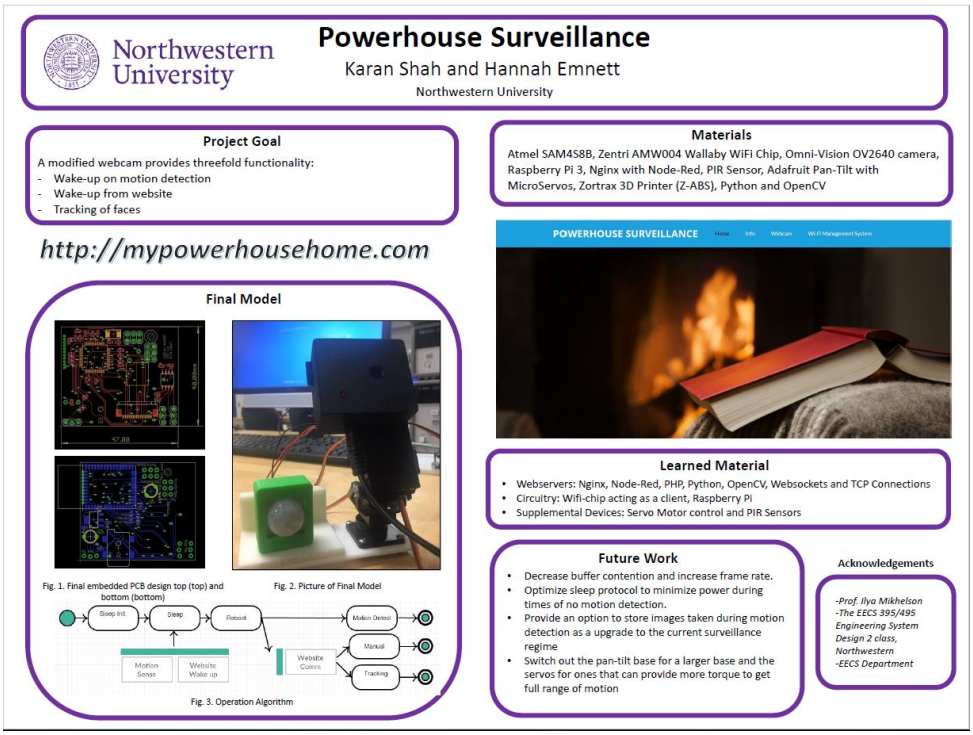


FIGURE 1: Poster utilized at the final exam fair.

Contents

List of Figures

Chapter 1

Introduction

Power House Surveillance first began in an EECS 495 classroom at Northwestern University. The original idea was to take a webcam that had been designed the quarter before and add motion sensing, a pan-tilt base that could be controlled remotely, and some image processing to enable a tracking mode. As before, the webcam features an Atmel SMART ARM-based SAM4S8B microcontroller, an AMW004 Wallaby Wi-Fi Chip, and an Omni Vision OV2640 as the camera. In the current edition of the camera, the microcontroller enters a sleep mode unless motion is detected. If motion is detected, it exits the sleep mode and sends the image to a website whose server is hosted on a Raspberry Pi. From the website, the user can choose to enter two submodes: Manual or Tracking. In Manual Mode, the user can manually control the pan-tilt base from the website. In Tracking Mode, the images are processed using a python script to identify the location of objects.

Chapter 2

Design Description

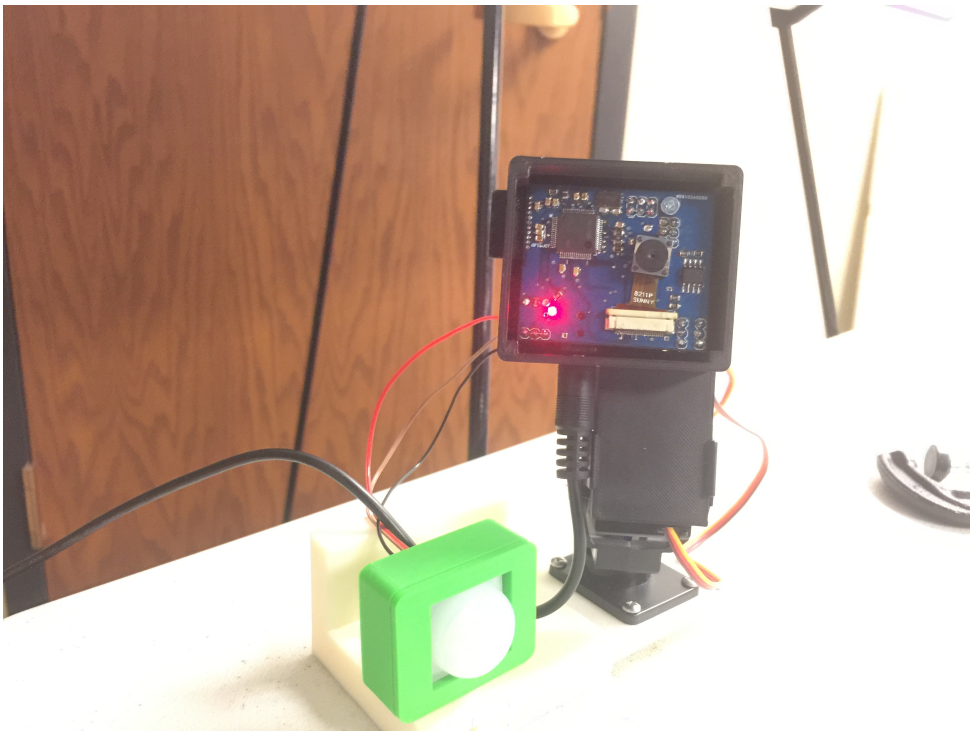


FIGURE 2.1: Final model of the embedded system with enclosure and base

2.1 System Overview

Essentially, the system starts with a bunch of initializations, including initializing the MCU/ Wifi-Chip pins and the micro-servos. The micro-servos start off in a zeroed position after the initialization. Next, the MCU goes into a sleep mode and enables the PIR sensor pin as an interrupt. This PIR sensor pin is also hooked up to a GPIO of the Wifi-Chip so that the MCU can be woken remotely. If motion is detected, the MCU re-initializes everything and starts transferring images. There is a TCP connection between the Wifi-Chip and the Node-Red server setup on the raspberry pi which is used to transfer commands to the pan-tilt base should manual mode be entered. There is additionally a websocket between the website and the Node-Red server to transfer commands from the buttons to the TCP connection.

From the motion detected state the user can enter manual or tracking state. The website was remotely accessible at http://mypowerhousehome.com. An image of the homepage of the website can be seen in Figure 2.2. A summarized materials list is as follows: Atmel SAM4S8B, Zentri AMW004 Wallaby WiFi Chip, Omni-Vision OV2640 camera, Raspberry Pi 3, Nginx with Node-Red, PIR Sensor, Adafruit Pan-Tilt with MicroServos, Zortrax 3D Printer (Z-ABS), Python and OpenCV.

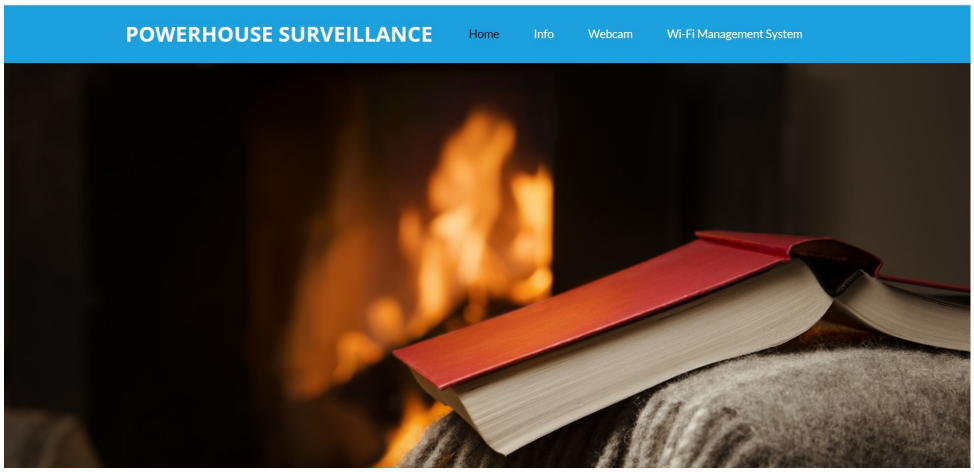


FIGURE 2.2: Homepage for the final website design.

2.2 Block Diagrams

The MCU main loop is composed of three different states. First, the MCU enters a non-blocking sleep-init state, where all of the sleep procedures are enabled and the PIR pin is set-up as a waking interrupt. Next, the MCU enters deep sleep. From the sleep state, the MCU can be woken up to enter motion detection mode. The first decision in motion tracking mode is whether or not to enter a separate state, manual or tracking mode, where it'll stop operating if motion isn't detected. In motion detect mode, once motion is not detected any longer, the MCU enters the sleep-init state. In manual or tracking mode, the MCU will never enter sleep-init state. Below in Figure 2.3 is a diagram of the basic MCU algorithm and following in Figure 2.4 is the actual state machine implemented.

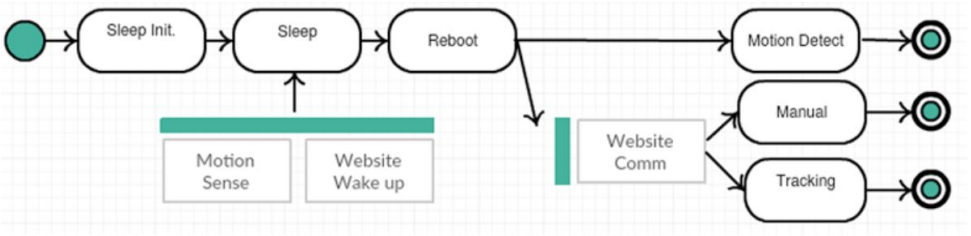


FIGURE 2.3: Graphical depiction of MCU algorithm.

The Node-Red flow utilized in the final edition of the project is shown in Figure 2.5. Essentially, a TCP connection is set-up between the Wifi-Chip and the Node-Red

```
while (1) { // Main loop
    switch (cur_state){
        case STATE_SLEEP_INIT:
            test_sleep_mode();
            cur_state=STATE_SLEEP;

        case STATE_SLEEP:
            break;

        case STATE_REBOOT_PIC:
            reconfigure_console(g_ul_current_mck, WIFI_USART_BAUDRATE);
            servo_config();
            delay_ms(10);

        case STATE_MOTION_DETECT:
            write_wifi_command("tcpc 10.106.6.149 1025\r\n",10);
            while (ioport_get_pin_level(PIR_PIN)) {
                if(start_capture()) write_image_to_file();
                write_wifi_command("read 0 20\r\n",10);
            }
            usart_write_line(WIFI_USART, "close all\r\n");
            pwm_channel_disable(PWM, PWML0_CHANNEL);
            pwm_channel_disable(PWM, PWML1_CHANNEL);
            cur_state= STATE_SLEEP_INIT;
            break;

        case STATE_TRACKING:
            pio_disable_interrupt(PIN_PUSHBUTTON_WAKEUP_PIO, PIN_PUSHBUTTON_1_PIN);
            while (track_mode) {
                if(start_capture()) write_image_to_file();
            }
            pio_enable_interrupt(PIN_PUSHBUTTON_WAKEUP_PIO, PIN_PUSHBUTTON_1_PIN);
            cur_state= STATE_MOTION_DETECT;
        break;

        case STATE_MANUAL:
            pio_disable_interrupt(PIN_PUSHBUTTON_WAKEUP_PIO, PIN_PUSHBUTTON_1_PIN);
            while (man_mode){
                if(start_capture()) write_image_to_file();
            }
            pio_enable_interrupt(PIN_PUSHBUTTON_WAKEUP_PIO, PIN_PUSHBUTTON_1_PIN);
            cur_state= STATE_MOTION_DETECT;
        break;
    }
}
```

FIGURE 2.4: State machine implemented on the SAM4S8B.

server. An additional websocket is set-up between the website and the Node-Red server. Not utilized in the final project but also included in the flow diagram is the code for posting the image and saving it to a remote location. In this edition of the project, all of the websocket code is directly available on the Node-Red server.

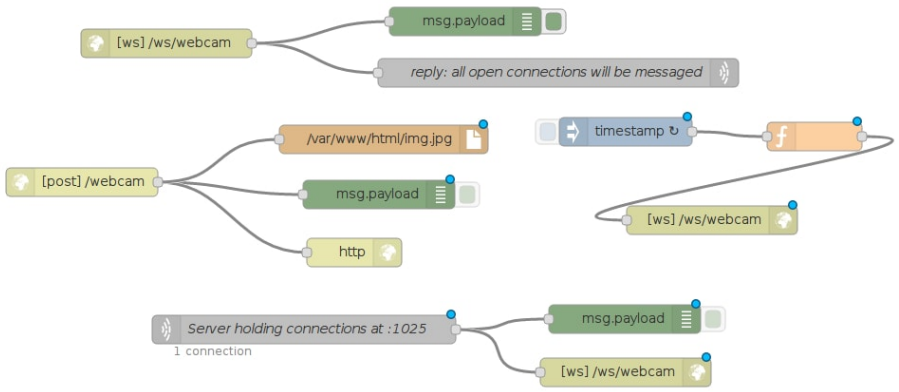


FIGURE 2.5: Node-Red flow utilized.

2.3 Algorithms and Code

A lot of programming was involved in the creation of this device. Besides those previously mentioned, code also had to be written in javascript to process information on the website. This websocket handled incoming requests from the Node-Red server and also output information so it could be transferred of the TCP connection to the MCU. Essentially, information was only transferred to the MCU after a new picture was uploaded to try to avoid contention issues. Additionally, if a button was clicked more than once in between picture uploads, the javascript stores that information and and sends it in one full package after the next upload of the image. A sample of this coding can be seen in Figure 2.6. The "doit" function is embedded into the html for each of the buttons. The servo motors in particular required a

```
function sendValueAll() {
    var val = 'dat' + pl + pr + tu +td;
    if (ws) { ws.send(val); }
    pl=0;
    pr=0;
    tu=0;
    td=0;
}
function doit(m) {
    if (m== "panright") {
        pr=pr+1;
    } else if (m== "panleft") {
        pl = pl +1;
    } else if (m== "tiltup") {
        tu=tu+1;
    } else {
        td=td+1;
    }
    if (ws) {ws.send(m);}
}
function doit1(m) {
    if (ws) { ws.send(m);
        document.getElementById('mode').innerHTML = "Manual mode engaged."
    }

}
```

FIGURE 2.6: Partial javascript code for pan/tilt functionality.

customized piece of code. Servo motors, unlike DC motors are commanded by specifying a position with pulse width modulation or PWM (essentially a square wave of varying duty cycle). In DC motors, the PWM directly correlates to speed (full duty cycle means maximum speed). For servo motors, the PWM directly correlates to a position over 180 degrees. Additionally, the PWM has to be updated with a frequency of 20ms and produce 1ms to 2ms pulses for 0 and 180 degrees.

The PIR sensor, which we initially considered to be difficult, actually ended up being quite easy once we acquired a sensory with all the supporting circuitry. The PIR sensor by itself outputs about a 10ms difference of amplitude in a high frequency sine wave. With some amplifiers and filtering however, a clean 0V and 3.3V can be directly fed into a digital input port to determine whether or not motion was detected. Additionally, the particular PIR sensor we bought can sense motion up to seven meters away which is ideal for our application.

The Omnivision camera is a low voltage image sensor that provides a full framed 8 bit image. The camera is connected to the MCU in parallel capture mode using PIODCCLK, PIODC data pins and PIODC enable pins. As it sends the data synchronously, it makes it suitable for our application which requires small size data at a faster rate.

The wifi chip contains inbuilt commands for posting and uploading an image either to an external website or the flash memory of the chip. In our implementation, we utilized both designs. Figure 2.7 demonstrates both functionalities. The commented portion represents uploading an image directly to the Wifi-Chip's flash memory where it can then be referenced from an external website on the same network. The uncommented portion involves posting the image to an external IP address where it can then be saved and referenced from a different server.

```
void write_image_to_file(void)
{
    if (len_image == 0) return;
    else {
        char file_create_string[100];
        //
        //wifi_file_delete_success = false;
        //usart_write_line(WIFI_USART, "fde image.jpg\r\n");
        //counts = 0;
        //while(!wifi_file_delete_success && counts<10);
        //
        //wifi_file_create_success = false;
        //sprintf(file_create_string,"fcr image.jpg %d\r",len_image);
        //usart_write_line(WIFI_USART, file_create_string);
        //
        //for (uint32_t jj=0;jj<len_image;jj++) {
            //usart_putchar(WIFI_USART, image_buf[soi_addr+jj]);
        //}
        //usart_write_line(WIFI_USART, "write 0 1\rh");

        sprintf(file_create_string, "hpo -o http://10.106.6.149:1880/webcam application/octet-stream\r\n");
        usart_write_line(WIFI_USART, file_create_string);
        sprintf(file_create_string, "write 1 %d\r", len_image);
        usart_write_line(WIFI_USART, file_create_string); //"write 0 2\rhi");//file_create_string);
        for (uint32_t jj=0;jj<len_image;jj++) {
            usart_putchar(WIFI_USART, image_buf[soi_addr+jj]);
        }
        usart_write_line(WIFI_USART, "hre 1\r\n");
        //usart_write_line(WIFI_USART, "list\r\n");
        usart_write_line(WIFI_USART, "close 1\r\n");
        usart_write_line(WIFI_USART, "read 0 10\r\n");

    }
}
```

FIGURE 2.7: HTTP POST and file create code for the AMW004.

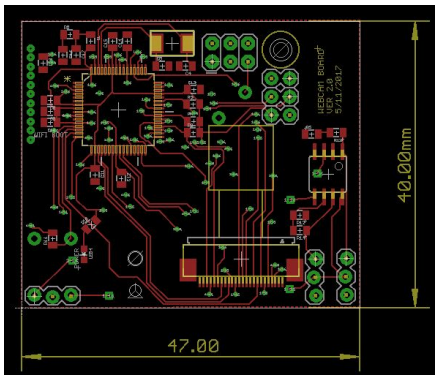


FIGURE 2.8: Eagle PCB design for the top of the board

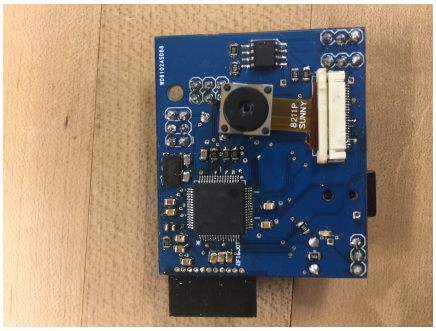


FIGURE 2.9: Final embedded model for the top of the board

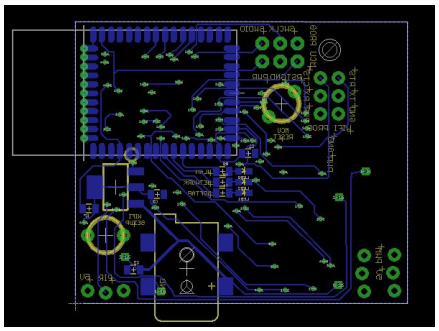


FIGURE 2.10: Eagle PCB design for the bottom of the board

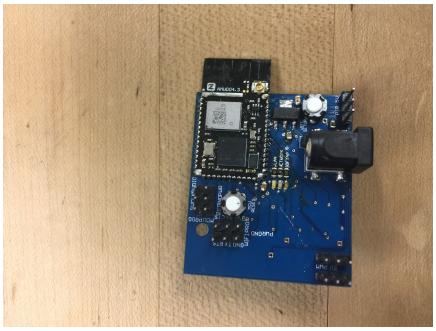


FIGURE 2.11: Final embedded model for the bottom of the board

The website was created using simple html and css code. The websocket script along with the image processing script is added to the webcam script. The websocket is used for opening the connection between the client and the server. The image processing script processes the image using opencv functions and viola jones algorithm. When the face is detected, the scripts draws a circle on the image and displays it back to the user.

2.4 PCBs

The printed circuit board (PCB) that we designed only had minor changes from the last iteration. First off, an amplifier was added for the control of the servos. Second, a connection was made for the PIR sensor and both servos. And lastly a GPIO of the Wifi-Chip was connected to the PIR pin to allow for remote wake-up. Pictures of the eagle and embedded designs for the top (Figure 2.8 and Figure 2.9) and bottom (Figure 2.10 and Figure 2.11)of the final PCB can be seen in the referenced figures.

2.5 3D Printing

In order to support the new design features, a new enclosure was designed that could support all the additional wires as well as mount the design to the pan-tilt

base. Additionally, an enclosure was made for the PIR sensor and as a way to mount the pan-tilt base to a surface. These enclosures connect the circuit board using one screw and the top is press fit for both the PIR sensors box and the final enclosure. A picture of the Onshape model for the final enclosure can be seen in Figure 2.12 and the link to the onshape file is shown here.

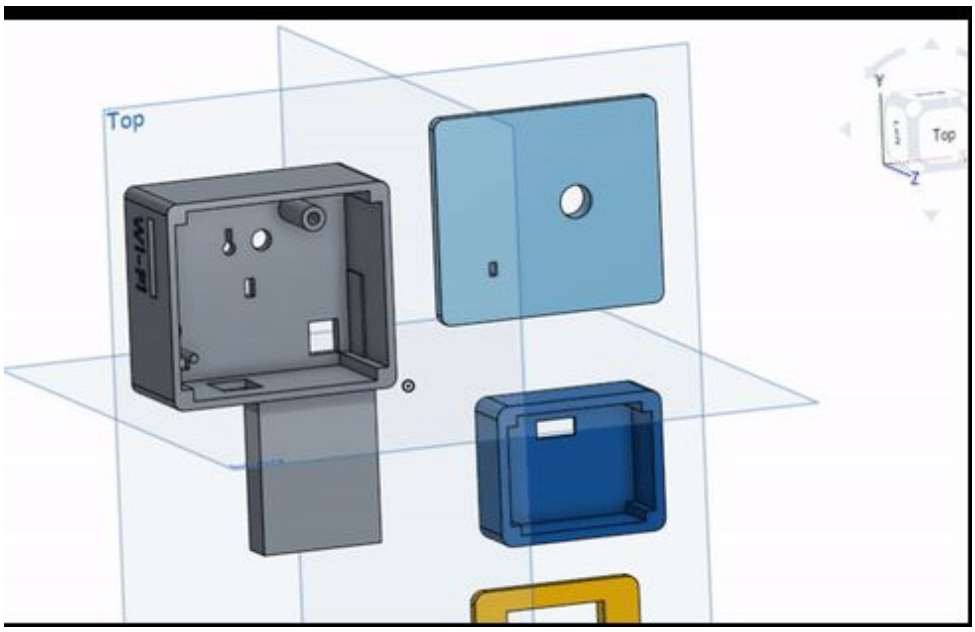


FIGURE 2.12: Enclosure for the embedded design and PIR sensor.

Chapter 3

Final Product

3.1 Initial Goal vs. Final Product

Our initial goals from the beginning of the quarter can be summarized as follows: have a camera that streams to a webserver remotely hosted on a Raspberry Pi (specific server not relevant), upgraded website to offer additional functionality, have a PIR sensor that can detect motion and only take photos when motion is detected, have a pan-tilt base that the camera is mounted to that can be controlled remotely, conduct some small image processing so that, if that mode is detected, the webcam can follows a face or a ball or a rat using its pan-tilt base. The final design as compared to the initial design accomplishes most of the initial goals with the addition of the following exceptions and add-ons: when the PIR sensor is not activated or there isn't a remote wake-up the MCU goes into a sleep mode that can help save power, the tracking software does not incorporate a pan tilt base but instead circles faces in an image.

3.2 Performance and Limitations

All of the components separately work flawlessly. The servos can be controlled precisely from the website, the PIR sensor can wake up the MCU and start/ stop motors, and the MCU can stream images to a remote website hosted on the Raspberry Pi. However, when we put all the components together using both a TCP connection and websocket, we suspect this is where we ran into issues. When you try to combine all the components, occasionally pictures get dropped or panning/tilting commands get dropped. In an effort to drop contention, the final design simply uploads the image to the flash of the Wifi-Chip and the website references this URL directly. In this case, no significant increase in performance was detected as this sometimes causes the website to reference an image that is not there. Other than this issue (which pretty much shadows over the whole project), the rest of it works pretty well.

Chapter 4

Challenges Encountered

Almost all of the separate parts of this project were learned material. In the final poster, we had a section, shown below as Figure 4.1, which is learned material. Although not an excuse for incurring delays and/or decreases in performance, it was definitely a contributing factor. The main challenge encountered was during the setup of the TCP connection and the websocket. This by far was the most challenging aspect of the project, simply because troubleshooting why something wasn't working by itself was extremely difficult. Unlike coding on the MCU, you cannot simply step through your code to see where errors are and where you could fix problems. Another problem arose in the simple setup of the Nginx server because similarly to when you are communicating through use of a websocket, with the exception of the error logs, it's difficult to pin point the error.

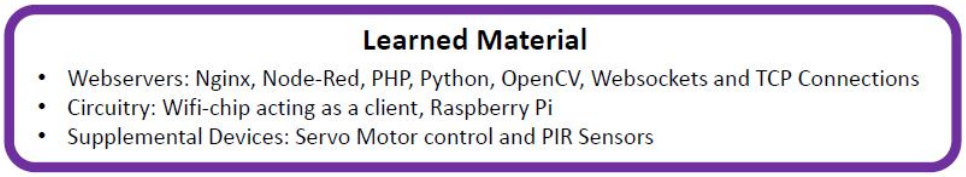


FIGURE 4.1: Summary of all the items learned by the team members in the design of this project.

Another challenge we encountered which was unanticipated was simply putting all the components together. In our gantt chart, the last two weeks were left to putting everything together. This left little time for troubleshooting issues (such as poor soldering jobs) and of course attempting to fix our main issue of contention.

Chapter 5

Planning and Organization

5.1 Gantt Charts

In an effort to ensure completion by the end of the quarter, a gantt chart (shown in Figure 5.1) was utilized to try to stay on schedule. For the most part, we stayed well on schedule with the exception of a couple delays. Everything to do with the internet essentially ended up causing delays, although these delays never were for more than a week. One change that should have been made to the gantt chart should have been doing the websocket first thing and testing with streaming the image and transmitting other information such as panning and tilting. What we did is confirm a websocket from the Raspberry Pi to the Wifi-Chip can transmit an image and that the website hosting on the Wifi-Chip can control the motors. Incremental full project testing could have saved a lot of headache in the final week of the project as well as probably turn out a better project.

	Start Date	End Date	Timeline	Status
EECS 395/495	Mar-28	Jun-5		
Define project goals and outcome objectives	Mar-28	Mar-31		Complete
Research components	Mar-30	Apr-4		Complete
Prepare order	Mar-30	Apr-4		Complete
Create PCB layout for motor connections	Mar-31	Apr-6		Complete
Create code for SAM4S to interface with Pan Tilt	Apr-6	Apr-13		Complete
Modify code for Rasp PI image processing	Apr-13	Apr-27		Complete
Modify code for SAM4S to stream to website with Pan Tilt	Apr-20	Apr-27		Complete
Modify pcb to incorporate servo and image processing/ motion detection	Apr-27	May-4		Complete
Modify website to interface with Rasp Pi and SAM4S	May-2	May-9		Complete
Put together embedded prototype (add servo connections to pcb)	May-4	May-11		Complete
Make revisions and design revised PCB	May-11	May-18		Complete
Keep perfecting code using breakouts	May-18	May-25		Complete
Put together final prototype	May-25	Jun-5		Complete
Prepare final presentation	May-30	Jun-5		Complete
		Burndown		

FIGURE 5.1: Picture of the gantt chart used during the project.

5.2 Communication Among Team Members

Communication between team members was never an issue. Besides the weekly planned meetings, we often met separately each week to go over goals we had between the current week and the next and to be clear about what work each of us was conducting. There was never a point where one of us thought the other was doing things throughout the duration of the project. The only issue that ever arose was when there was an expectation that a part of the project would only take one

week but ended up taking more. When that happens it of course sets back the next person's progress by a whole week if they are waiting to utilize that portion of the project. But that as well was always clearly communicated.

5.3 Splitting Tasks Among Team Members

As the structure our project featured with the gantt chart having incremental tasks to be completed, it was easy to split tasks up amongst team members where one of us would work on the image processing portion and the other one would work on motor control. This of course caused problems later on when everything needed to be combined but as far as splitting up jobs no issues were encountered. Of course another benefit is that our project was made up of a bunch of small components. Had we had a more streamlined project this would have caused more of an issue. Even still, it became difficult toward the end of the project as both of us would need to use the Raspberry Pi but of course you can't work on it both at the same time. Either that or one person would have to wait till the other completed a portion of the project before they could work on their section.

Chapter 6

Conclusion: Karan

6.1 What You Learned

From this course, I have learned more about embedded system. I learned about the different configurations that can be used for designing and establishing connection between server and client. One of the main things was learning http post and web-socket communication along with node red for configuration of server. The whole course was a learning curve to establish a web server which can connect with wifi chip using http post or websocket. In the end, we used TCP/IP and websocket for connection purposes.

6.2 What You Would Do Differently If You Could Start Over

If would be given the option of starting over, I would first configure the server and establish a connection between the server and the wifi chip. My preference would be to use Node red from starting of the course as it is easier to use and have many resources for reference. After establishing the server, I would focus on the type of connection that would be suitable for the application and then focus on the hardware for the project.

6.3 Possible Next Steps

The possible next steps would be decrease the buffer contention that is taking place between the TCP connection and http post. The other step is to increase the frame and optimize the code to increase energy efficiency when no motion is detected. The future direction can also consist of providing an option to store the captured image.

Chapter 7

Class Feedback: Karan

Questions to Answer: Did you learn as much as you hoped to in this class? Do you have any suggestions for improvement of the class format or structure to increase learning? Sum up your thoughts to this project, the class, and your overall experience

My learning experience for the course was pretty high. The things that I wanted to learn were covered in one course. More freedom should be given to the students in such a way that can choose to incorporate the previous quarter embedded system or develop something new. Th e project was a difficult and in the end, we managed accomplish our goals. The class structure was proper and fun. The overall experience was good and I would surely recommend it to other students who have interest in embedded system design.