Northwestern University

McCormick School of Engineering

Department of Electrical and Computer Engineering

ELEC_ENG_326: Electronic System Design - 1

Winter 2023

Webcam

Final Report

Adheik Dominic

Date: March 17th, 2023

INTRODUCTION:

In this project, we aim to create an embedded webcam using the Atmel SAM4S microcontroller and ESP32 Wi-Fi chip. The SAM4S microcontroller will interface with the camera module and capture video frames. The ESP32 Wi-Fi chip will be responsible for transmitting these frames over the internet to a website. We will use the JPEG compression standard to compress the video frames and transmit them over the internet in real-time. Additionally, we will design and 3D print a custom enclosure for the webcam that will house all the components and provide adequate protection.

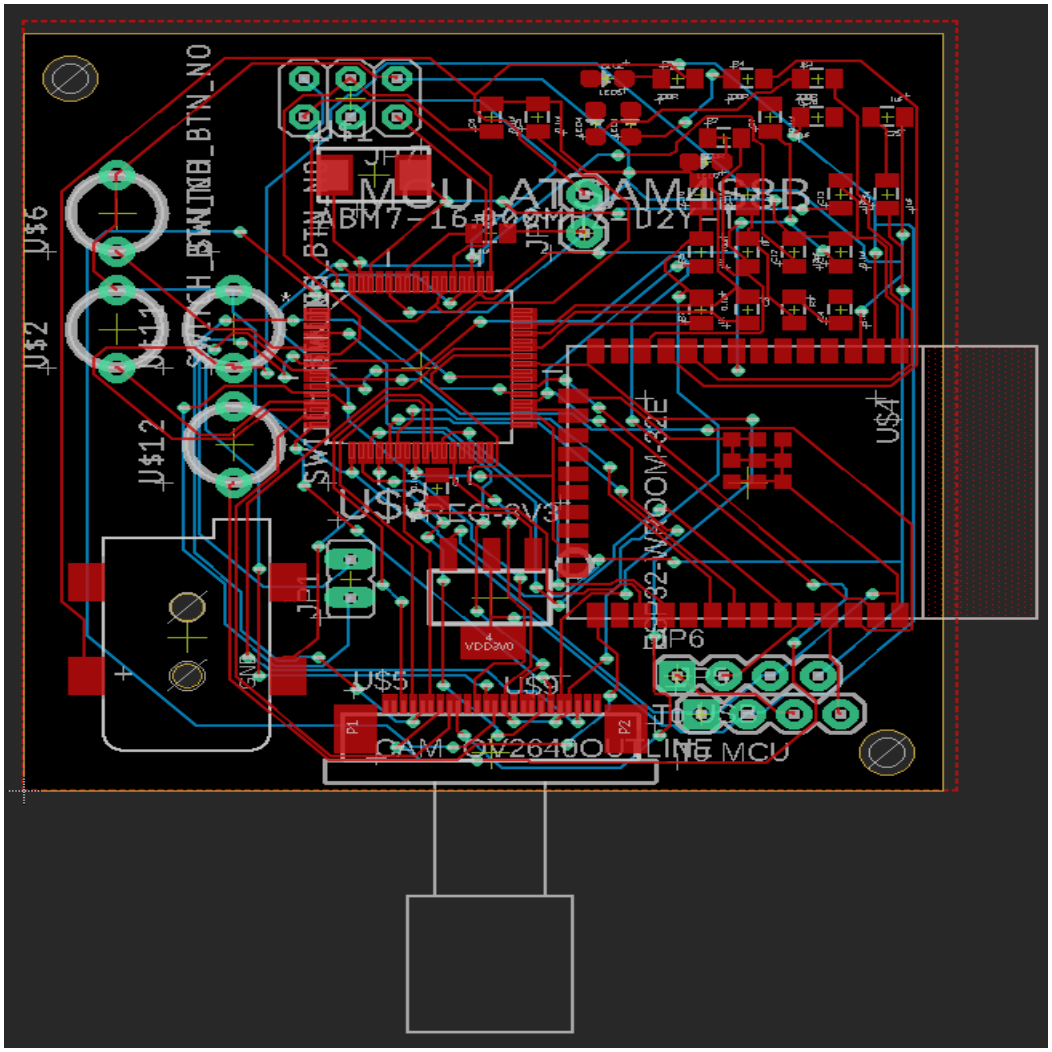
DESIGN PROCESS:

PCB DESIGN:

1. Orientation: Make sure all components are placed in the correct orientation. Check the datasheets of the components to determine the correct orientation, and make sure to align the markings on the components with the corresponding markings on the PCB footprint.
2. Mechanical placement: Ensure that there is enough space between components to allow for proper assembly and operation. Also, make sure there is enough clearance for any heat sinks or other mechanical parts that may be attached to the components.
3. Signal integrity: Place components in a way that minimizes the length of traces and the number of vias between them. This helps to maintain signal integrity and reduce noise and interference.
4. Power distribution: Place decoupling capacitors close to the power pins of each component to minimize power supply noise. Additionally, make sure the power traces are wide enough to handle the required current.
5. Thermal considerations: Place components in a way that allows for proper heat dissipation. Make sure components that generate heat, such as voltage regulators and power amplifiers, are placed in areas with good airflow and/or heat sinking.
6. Design for manufacturability: Place components in a way that is easy to manufacture. Avoid placing components too close to the edge of the board or too close to each other to allow for proper assembly and testing. Also, try to keep the board as compact as possible to reduce costs.

I feel the above 6 should be followed to make a PCB design using eagle. Coming to my design I wanted to place all the components on the top layer of the PCB so that it would be easier to

route. We had placed a barrel jack in the corner. I tried to group and place all the LEDS, Capacitors and resistors together as much as I could. I placed the camera on the opposite edge of the SAM4S so that the camera is coming out a bit. I specifically went for this approach as I wanted everything on the top but end of the day it is up to the person. So below is my PCB please see how similar components and grouped and placed together this actually makes the routing much easier.



FIRMWARE:

Main:

During initialization, you will need to configure and initialize various components of the system, including the clock and board definitions, timer, WiFi USART and SPI, GPIOs, camera, and WiFi connection. To ensure proper operation, you will also need to check for a successful connection to the WiFi network and listen for the "provision" pin. Additionally, you will need to send a "test" message to the WiFi module and wait for a "SUCCESS" response. If this response is not received, the system will need to reset the WiFi module and try again. Once the initialization is complete, the program will enter the loop section. In this section, the program will check for any provisioning requests and respond accordingly. If the network is available and clients are connected, the program will take a picture. Finally, if the picture is taken successfully, it will be transferred to the ESP32.

```
int main(void)
{
	sysclk_init();
	board_init();
	configure_tc();
	configure_usart();
	configure_spi();
	configure_wifi_flag();
	configure_client_flag();
	configure_provision();
	wifi_init();
	init_camera();
	configure_camera();
	wifi_reset();

	while(!clients_connected){
		if(provisioning_requested){
			usart_write_line(BOARD_USART,"provision\r\n");
			provisioning_requested=false;
		}
	}
	while (true) {
		usart_serial_write_packet(BOARD_USART, "test\r\n", 6);
		delay_ms(100);
		char response[10];
		uint32_t length = 0;
		usart_serial_read_packet(BOARD_USART, response, sizeof(response));
		if (length > 0 && (response == "SUCCESS\r\n") == 0) {
			break;
			} else {
			wifi_reset();
			delay_ms(10000);
		}
	}

	while (true) {
		if (provisioning_requested) {
			usart_write_line(BOARD_USART,"provision\r\n");
		}
```

```
            if (ioport_get_pin_level(wifi_connected) &&
ioport_get_pin_level(clients_connected)) {
                    if (start_capture()) {
                            write_image_to_web();
                    }
            }

      }
}
```

Wifi:

The first function, "write image to web", transfers an image from the SAM4S8B to the ESP32 by configuring the SPI interface, issuing a command, and waiting for the ESP32 to signal the completion of the transfer via a GPIO pin. The second function, "write wifi command", writes a command to the ESP32 and waits for an acknowledgment or timeout by checking a global variable.

We had used three interrupts like the one below.

```
static void wifi_comm_handler(uint32_t uil_32, uint32_t uil_33){

      unused(uil_32);
      unused(uil_33);

      wifi_comm_flag = true;
}

void configure_wifi_comm_pin(void)
{
      pmc_enable_periph_clk(WIFI_COMM_PIN);
      pio_handler_set(WIFI_COMM_PIO, WIFI_COMM_ID, WIFI_COMM_PIN_MSK, WIFI_COMM_ATTR,
wifi_comm_handler);
      NVIC_EnableIRQ((IRQn_Type)WIFI_COMM_ID);
      pio_enable_interrupt(WIFI_COMM_PIO, WIFI_COMM_PIN_MSK);

void write_wifi_command(char* comm, uint8_t cnt){
      usart_write_line(BOARD_USART, comm);
      counts=0;
      while(cnt>counts && response_flag){

      }
      response_flag = 0;
}

void configure_wifi_comm(void){

      pio_handler_set(WIFI_COMM_PIO, WIFI_COMM_ID, WIFI_COMM_PIN_MSK,
      WIFI_COMM_ATTR, wifi_comm_handler);

      /* Enable PIO controller IRQs. */
      NVIC_EnableIRQ((IRQn_Type)WIFI_COMM_ID);

      /* Enable PIO interrupt lines. */
      pio_enable_interrupt(WIFI_COMM_PIO, WIFI_COMM_PIN_MSK);
```

```
	wifi_comm_flag = false;
}
```

This is how we defined our “write_image_to_web” function as you can see we are using spi_prepare_transfer to write images to the web. We got the other functions from the SPI example.

```
void write_image_to_web(void){
	spi_prep_transfer();
	uint32_t leng = img_len
	if(leng > 0){
		char buffer[50];
		sprintf(buffer, "image_transfer %U", leng);
		write_wifi_command(buffer, leng)
	}
}
```

Below was our “write_wifi_command” function you can see, we are using a char and a count to write this.

```
void write_wifi_command(char* comm, uint8_t cnt){
	usart_write_line(BOARD_USART, comm);
	counts=0;
	while(cnt>counts && response_flag){

	}
	response_flag = 0;
}
```

Camera:

This is our “start_capture” function which captures an image after VSYNC rising edge and returns its length. Returns 1 if successful (nonzero length), 0 on error.

```
uint8_t start_capture(void)
{

	/* Enable vsync interrupt*/
	pio_enable_interrupt(OV7740_VSYNC_PIO, OV7740_VSYNC_MASK);

	/* Capture acquisition will start on rising edge of Vsync signal.
	 * So wait g_vsync_flag = 1 before start process
	 */
	while (!g_ul_vsync_flag) {
	}

	/* Disable vsync interrupt*/
	pio_disable_interrupt(OV7740_VSYNC_PIO, OV7740_VSYNC_MASK);

	/* Enable pio capture*/
	pio_capture_enable(OV7740_DATA_BUS_PIO);

	/* Capture data and send it to external SRAM memory thanks to PDC
	 * feature */
	pio_capture_to_buffer(OV7740_DATA_BUS_PIO, g_p_uc_cap_dest_buf,
```

```
                100000 >> 2);

        /* Wait end of capture*/
        while (!((OV7740_DATA_BUS_PIO->PIO_PCISR & PIO_PCIMR_RXBUFF) ==
                PIO_PCIMR_RXBUFF)) {
        }
        picture_taken=true;
        /* Disable pio capture*/
        pio_capture_disable(OV7740_DATA_BUS_PIO);

        /* Reset vsync flag*/
        g_ul_vsync_flag = false;

        if(img_len != 0){
                return 1;
        }
        else{
                return 0;
        }
}
```

The “find_image_len “ function finds image length using JPEG protocol, returning 1 if successful (found "end of image" and "start of image" markers), 0 on error.

```
uint8_t find_image_len(void) {
        uint8_t i = 0;
        uint8_t x;
        uint8_t start_of_image = 0;
        uint32_t bufpos = 0;

        // Read bytes from buffer until end of image marker is found
        while (!start_of_image) {
                x = g_p_uc_cap_dest_buf[bufpos++];

                if (x == 0xFF) {
                        x = g_p_uc_cap_dest_buf[bufpos++];

                        // Check for start of image marker
                        if (x == 0xD8) {
                                start_of_image = 1;
                                } else if (x == 0xD9) {
                                // End of image marker found without start of image marker
                                return 0;
                        }
                }
        }

        // Read bytes from buffer until end of image marker is found
        while (1) {
                x = g_p_uc_cap_dest_buf[bufpos++];

                if (x == 0xFF) {
                        x = g_p_uc_cap_dest_buf[bufpos++];

                        if (x == 0xD9) {
                                // End of image marker found
                                return 1;
```

```
                        } else if (x == 0xD8) {
                        // Start of image marker found before end of image marker
                        img_len = 0;
                        } else if (x == 0x00) {
                        // Skip over any 0x00 byte (used for padding)
                        continue;
                        } else {
                        // Read the length of the segment and skip over it
                        uint16_t len = ((uint16_t) g_p_uc_cap_dest_buf[bufpos++] << 8)
| x;
                        img_len += len - 2;
                        for (i = 0; i < len - 2; i++) {
                                g_p_uc_cap_dest_buf[bufpos++];
                        }
                }
                } else {
                // Not a marker byte, skip over it
                img_len++;
        }
    }
}
```

WEBSITE:

For our webcam we are using the website to stream the image from the webcam in real time. The website is made of three components: HTML, CSS and JAVASCRIPT. Our website is made of 3 pages – Info, Index and webcam stream.

The index page tells us about the project and is the welcoming website. You can see the webcam image and a little about the project as seen in the image below. After auditing the website, we got a 100 on performance

The second page is the info page which tells you about the team and the people involved in the project. You can also find contact information of every member as seen below. The auditing results of the info page gave 74 on performance.

The third page is the webcam. It will stream the webcam image from our camera. It also has a timestamp and a button to start and stop the webcam. The log section will tell about the camera connection. The final audit results are 100.

The website is based on HTML; however, it is styled using CSS. For the webcam page, we used JavaScript to make it dynamic. It was needed to stream and show the timestamp in real-time.

We added a navigation bar on every page to make it easy to access through every page.

```html
<!DOCTYPE html>
<html lang="">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="info.css"/>
    <title>Our info</title>
</head>
<body>
    <div class = "main ">
        <nav>
            <ul>
                <li><a href = "index.html" target = "_blank">Index</a></li>
                <li><a href = "webcam.html" target = "_blank">Webcam</a></li>
                <li><a href = "/edit" target = "_blank">Edit</a></li>
            </ul>
        </nav>
    </div>
    <h1>Meet the team!</h1>
    <h2>Shivi Shrivastava</h2>
    <h3>Contact: shivishrivastava2024@u.northwestern.edu</h3>
    <h4>Electrical Engineering</h4>
    <img src="shivi.jpg" width = "200" alt = "Unable to display image">
    <h2>Adhiek Dominic</h2>
    <h3>Contact: adheikdominic2023@u.northwestern.edu</h3>
    <h4>Computer Engineering</h4>
    <img src="adhiek.jpeg" width = "200" alt = "Unable to display image">
    <br><br>
</body>
</html>
```

Figure: Webcam page

```
function onOpen(evt) { // when handshake is complete:
    writeToScreen("Connected.");
    b.innerText = ("Stop Webcam")
    b.title = "Click to stop webcam"
    b.disabled = false;

    buttonClicked = false;
}

function onClose(evt) { // when socket is closed:
    writeToScreen("Disconnected. Error: " + evt);
    b.innerText = "Start Webcam"
    b.title = "Click to start webcam"
    b.disabled = false;
```

Figure: Button functionality

```
var date = new Date();
document.getElementById("timestamp").innerHTML = ("Timestamp: " + date.toString());
// Get the image just taken from WiFi chip's RAM.
```

Figure: Timestamp

3D-ENCLOSURE:

The PCB designed for the webcam needs an enclosure for it to be put to use. We used OnShape to create the 3D design. The design included a box for the PCB and a lid to cover the hardware. The box was a 43X43 box and 28 mm depth for our 40X40 PCB. All the walls of the box were 1 mm thick.

The PCB has two holes for a 2.5 X 12 mm dimension screw. There are two standoffs for the screws in the PCB. There is an 11X25 mm cutout for the barrel jack. The cutout is all the way to the top for easy insertion. There is another cutout of 20x7 for the wifi antenna.

The cover (lid) of the box is also a 43X43 mm lid with a 2 mm thickness. A few other cutouts will include the 4 buttons (2 on each side), power LED, indicator LEDs, programming header and USB to UART connections.

We have also added a hole at the back of the box to clamp the webcam on a wall or any other surface. We added a hook on the lid to get attached to the box so that it is easy to remove the cover if the user wishes.

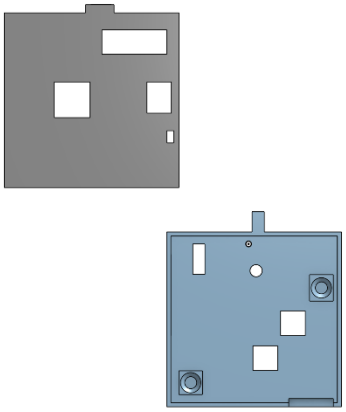


Figure: Front view

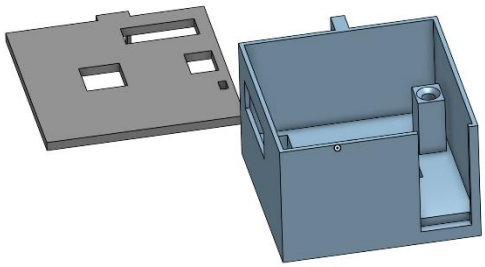


Figure: Side view

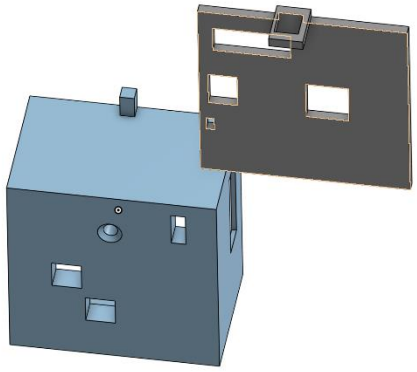


Figure: Back view

Discuss what you learned through the design process.

** What challenges did you encounter?*

So, we had an issue with our PCB design and our firmware. In the PCB design we had two RX and TX pins connected to each other and this was because we used auto-routing in EAGLE. We also had a lot of misconnections. Also, surprisingly two of our ESP32 pins were connected with each other. So, we had to use a knife to cut it and then we had to short two diodes to short them in order for it to work. The best part is sometimes your board does not work and you don't even know why because from your aspect everything is correct. Sometimes your firmware works but it does not work on the board. While coding the firmware, we faced a lot of issues. Firstly, we did not configure the SPI and the GPIO pins. Start capture function was not working properly and the find image length function was not returning the size starting from the FF D8 marker and was throwing random images We also had an issue with the write image to web function as it was not using the spi transfer properly. Finally, our main function was not working properly and we were calling functions in the wrong manner. Another challenge was mostly in the website design. It was the first time we were designing a website and we did not know a lot about HTML, CSS or JavaScript

For the enclosure, there were very small mismatches between the design and the actual board. For example, the button cutout was not big enough, the standoffs did not align with the actual screw hole on the PCB or the board was a little bigger than expected. This led us to print the enclosure thrice before we got the correct one. I had to calculate the design parameters again and calculate mismatches from the printed enclosure to get the correct one.

** How did you overcome these challenges?*

So first for the PCB design issue we had tried cutting it. That did not work with the first two boards. So, for the third board we cut it and after soldering everything the POWER LED was still not working, we were able to find the short but that still did not solver our issue. We finally removed the ESP32 completely and tried to resolder it but this the pads were damaged so we had tried to resolder them but that failed. We finally borrowed professor's board but that did not work as well. Our board was apparently getting a higher than 3.0 V voltage and this we had to resolder this and we finally just used our breakout boards for the demo. So, most of the firmware problems were solved after initializing the GPIO and the SPI pins correctly. Then once we defined the "find image len" function correctly so we found the first marker FFD8 and incremented it to keep returning the bytes until we find the last marker FFD9. This function gave us a perfect image. We fixed "write image to web" function by using the "spi_prepare_transfer" from the example project and that worked. We also changed the image buffer from a pointer to an array and this fixed our "start_capture" function. Our main loop needed some changes and we used two while loops and then checked to see if WIFI and client pins were high in order to start capturing images and write them to the web. Website problems were only solvable because of the immense help from the professor and the peer mentor and TA. The Enclosure was learnt through mistakes as I feel the mistakes we made while making the 3D design was what gave us ideas for our final design.

What would you change if you had the chance to start over? Mention what you would change in:

** Your approach to the problem.*

So firstly, I would not use auto routing in EAGLE CAD because this ends up in making unnecessary connections and that are hard to solve once the PCB is manufactured. These will keep causing shorts and we will not be able to solve them easily. So, I would have definitely used the normal routing and made sure I do them correctly and triple check my connections. For the firmware coding. I would first make a skeleton code and estimate how to call all the functions and in which order. This time around I will make sure that I initialize all the SPI and GPIO pins correctly. Before, I deep dive into the coding I will check if the image is being captured properly and is written to the web correctly. Website should be checked properly and we should make sure our html and JS files work properly. So, I would preferably next time start coding much earlier. As for the enclosure we need to make sure the design is as accurate as it is.

** Your actual design.*

When designing a PCB, it's important to carefully consider the placement of each component to ensure proper functionality and manufacturability. This includes making sure each component is oriented correctly, providing adequate space for assembly and operation, optimizing signal

integrity by minimizing trace length and vias, placing decoupling capacitors close to power pins, accounting for heat dissipation, and designing for manufacturability by avoiding tight spacing or board edge placements. By taking these factors into account, you can create a well-designed PCB that is optimized for performance, reliability, and ease of manufacturing. When writing an embedded program, it is important to plan and design the program before writing any code. This will help ensure that the program is structured, modular, and scalable. Using more interrupts and for thoroughly testing and debugging the program is essential to ensure it works as intended. By following these guidelines, you can create a well-designed, optimized, and reliable embedded program that meets the needs of your application. To make a good website I would design a visually appealing user interface that is easy to navigate, with clear options for accessing the webcam feed. I would ensure that the website is optimized for fast loading times and responsive design. To create a good webcam enclosure in Onshape, I would start with clear design requirements and choose materials appropriate for the intended use. Pay attention to the webcam's size and placement, ensuring proper ventilation and access to any necessary ports. I would then consider adding features such as mounting options, cable management, and protection from dust and debris. Test the enclosure for fit and functionality, making any necessary adjustments before finalizing the design.

TEAMWORK

Did you contribute fairly to your team effort? Do you think that you did an unfair amount of work, either too much or too little, as compared to your partner?

Yes. My partner was of immense help, we equally distributed the team load and did all the tasks properly. The final few tasks like the firmware, website, soldering and enclosure were equally distributed among us and we were able to finish everything. However, since my partner did not know embedded coding and I did the firmware completely alone I felt I could have a little more help on the firmware and coding the firmware alone was a tad bit difficult.

LEARNING

–Why did you take this class and what did you hope to learn?

I took this course and class because I love hardware electronics and embedded programming. I wanted to learn PCB design, soldering and CAD. I also wanted to learn embedded coding at a more advanced level.

–Did you learn as much as you hoped to in this class?

This class was truly amazing and I learned way more in this class then many of the other classes I have taken in Northwestern. We learned a lot in this class from PCB design, EAGLE CAD, Soldering, making websites, 3D programming and embedded programming.

– Were there any topics that you hoped to learn but did not?

Nope I learnt all the topics that I had hoped to learn in this class.

– Do you have any suggestions for improvement of the class format or structure to increase learning?

No, but I would highly recommend making this a 400 level course considering the amount of workload that is involved. The course structure is perfect however. I really liked how all the 8 tasks finally lead to building a webcam.

– Do you think the workload was too high, appropriate, or too low? Please elaborate on your response. If too high, could it be lowered without decreasing the amount learned? If too low, what would you like to see added?

I personally felt the workload was a bit high in comparison to other classes, I would have easily spent around 15 hours every week just in the lab. I feel instead of making the PCB design towards the end it would be better if the PCB was made in the beginning and we have enough time to make changes or get a new PCB if ours is damaged.

CONCLUSION:

This project was quite honestly like a movie. It was an absolute pressure to go through the highs and lows that I faced in this project. The satisfaction that you get when you successfully complete a task is an indescribable feeling. I learnt so much in this course. There is almost nothing that you do not learn in this course if you love hardware and embedded programming.

The class structure was brilliant and we were always offered help when we needed it. My overall experience is that this was one of the best classes I have taken so far and I would gladly recommend this class to anyone and everyone.