PORTABLE THERMAL PRINTER

By

Gally Huang (ghuang23)

Jason Liu (jliu246)

Kevin An (kqan2)

Design Document for ECE 445, Senior Design, Spring 2023

TA: Shao Hanyin (hanyins2)

Team 29

Contents

1. Introduction

1.1 Problem

One of the biggest problems surrounding frequent travelers is the issue of portability. Items that are carried along have limits on their weight, cannot consume too much space, and must not compromise on quality. A target area that has been identified by the Hewlett-Packard Company (HP) lies within the commercial printer industry. Printers have remained relatively unchanged over time with respect to other technologies that have shifted towards more portable means. As such, they remain inconvenient for travelers who need to quickly print items on the go. HP has identified a potential entry into the portable printer market to remain competitive in this industry and find new methods for company innovation.

1.2 Solution

Our solution is a portable thermal printer, a system that receives wireless instructions for printing on thermal paper. Users will be able to upload images from their phones or computer that this system can fetch and print.

We will use a field-programmable gate array (FPGA) to implement our hardware solution because they can stand in place for a real-world application-specific integrated circuit (ASIC) and eventually be developed commercially in an ASIC. It will be utilized as the base of the project. Additionally, we need to have a way to print, so we will be using the internals of a thermal printer along with a Wi-Fi enabled microcontroller unit (MCU) to handle the wireless functionality. Finally, we will be creating our own custom input/output shield (I/O shield) for the PCB that has the subsystems listed further down on top of it.

1.3 Visual Aid

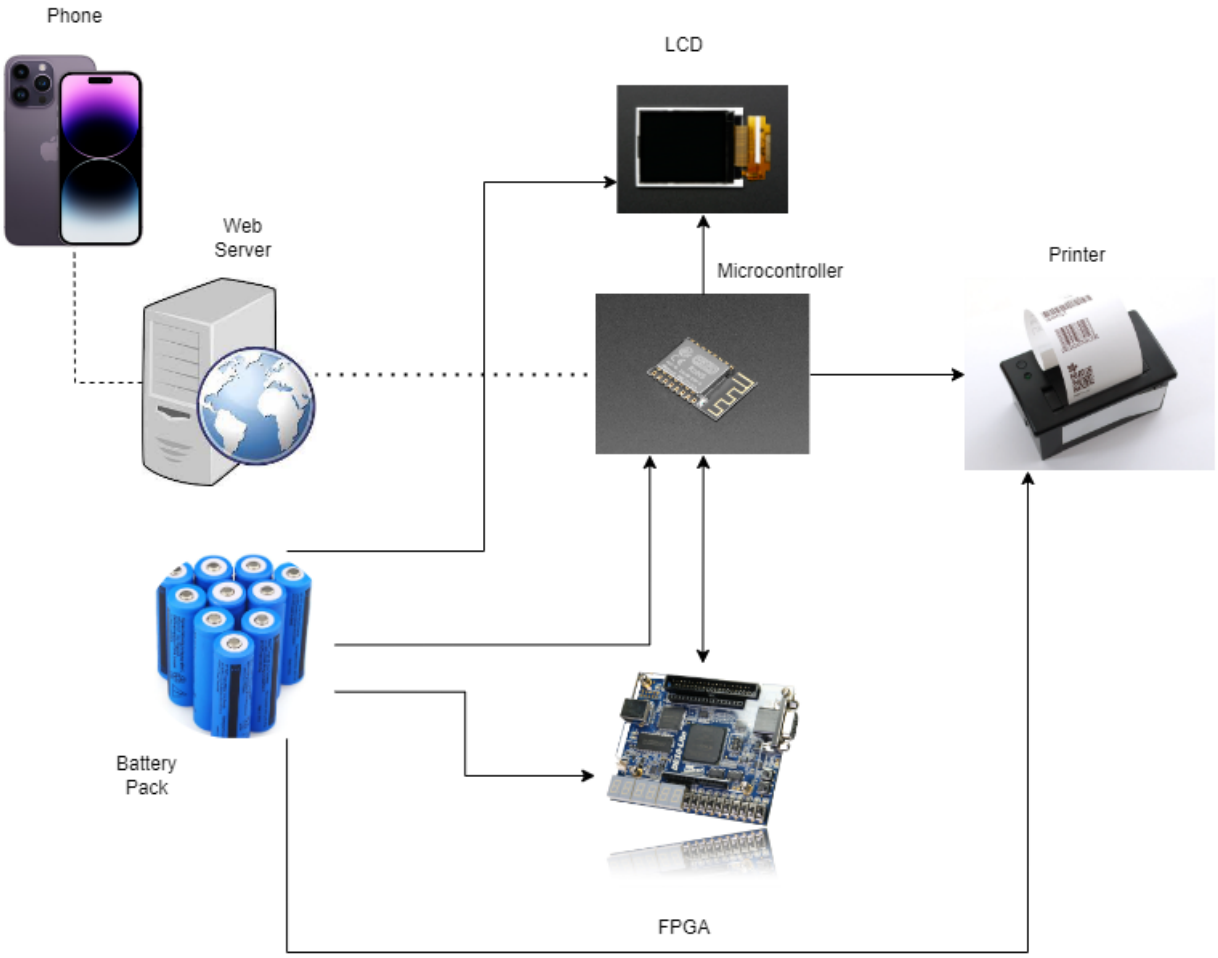

Figure 1. A high-level visual representation of the general usage of the Portable Thermal Printer.

1.4 High-Level Requirements

1. The device design is portable. It should be able to get the user-uploaded image data wirelessly and accurately from a server to the embedded MCU. It should sit as a small footprint of at most 12"x12" to fit comfortably within a suitcase, allowing for easy transportation.
2. The device itself should also be completely powered by batteries, having an average (if not worst case) battery life of ideally 1.5 or more hours.
3. The start to end time, between user upload and completing the printing, should be within 20 seconds to not consume too much time for the user.

2. Design

2.1 Block Diagram

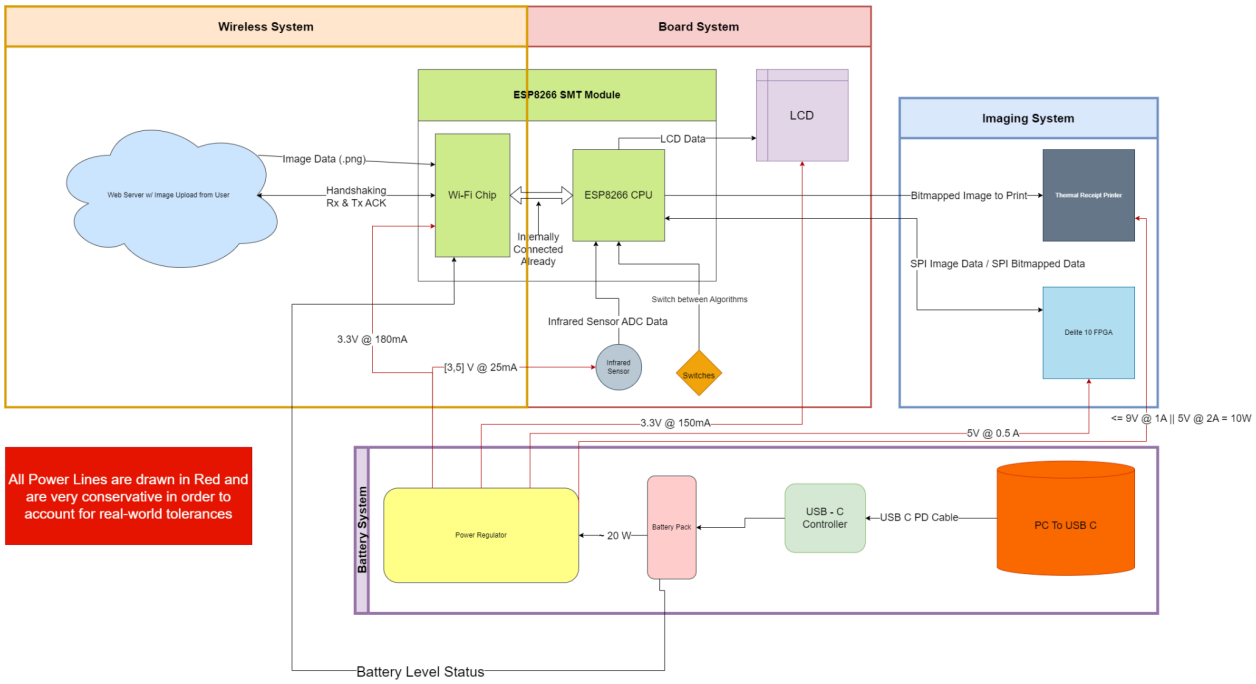

Figure 2. Portable Thermal Printer system block diagram.

2.2 Wireless Subsystem

2.2.1 Description

The purpose of the Wireless Subsystem is to allow the system to wirelessly connect between a server (can be locally hosted on a computer or on the cloud), a user, and the ESP8266 ESP-12F MCU. The benefits of this subsystem add portability for the product and a more "modern" feel for the user, reducing the need for excessive cables and clutter.

There will be a simple web page backed by the server that a user can interface with and upload an image to and upon which, the user can request a connected printer to print the uploaded image. The server will send data to the MCU through a wireless socket connection between the MCU and the server. After receiving this data, the MCU will further process and deliver it to other respective subsystems.

2.2.2 Requirements and Verification

Table 1: Wireless Subsystem R&V

Requirements	Verification
• Connected users can upload an image successfully quickly to the server. • Images will need to be delivered to the server within 5 seconds of upload to reduce the amount of overall delay for the user.	• Starting with activating the server, at the time of user upload, we will manually check the time using a stopwatch (i.e., with a cellphone) to determine that the data was received on the server within 5 seconds. • Upon a successful user upload, the image that is delivered to the server should appear in the host computer's local storage, as in the directory that the server code is active on. This will allow us to verify that the necessary data has been received in full within the given time and that the custom API is successful in this requirement. • We will repeat this for many scenarios, such as having multiple users on the server simultaneously, varying image sizes and resolutions, and at different distances from the host computer.
• The ESP8266 ESP-12F microcontroller requires 3.0-3.6 V continuously from the power subsystem for operation.	• Ensure the power system is active without having the MCU soldered in place. We will use an oscilloscope to measure the voltage across the Vin and GND pins on the breakout PCB to ensure that the voltage range is safe for the MCU operation. • Record the voltages throughout many different points of operation of the battery life to ensure that the Vin does not fluctuate outside of the 3-3.3 range. Any higher than 3.6V will damage the MCU.
• The ESP8266 ESP-12F microcontroller can receive asynchronous image data from a WebSocket client.	• Due to the limited resources on board, we will need to send the image data to the MCU client as a base64 encoded string. • At every WebSocket event initiated by the server, we will print out the event

	JSON data in the Arduino IDE console and manually verify that each string(s) matches the corresponding image base64 encoded through a separate Python script.

The ESP8266 ESP-12F microcontroller is a low-cost MCU that will be embedded on the custom PCB (we will design this as an I/O Shield for the system's FPGA). With respect to the wireless functionality, it is responsible for allowing the printer system itself to stay wireless, as it has a built-in Wi-Fi microchip, enabling simple connection to an application server (discussed below). With the MCU acting as a client to an application server WebSocket, it can listen for a request event made by the server, where the encoded image data is sent through the connection.

This MCU will then be responsible for serially delivering the image data to a buffer for the FPGA to further process towards printing, taking advantage of the FPGA for hardware acceleration in implementing DSP algorithms (such as the Floyd-Steinberg dithering algorithm) to speed up the image manipulation and cleaning processes. It will also send diagnostic data about the state of the current processes to an LCD display, such as about current battery status, ready for printing acknowledgement, and printing completion/failure statuses.

Hardware Paramaters	Peripheral Bus	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	5x5mm
	External Interface	N/A

Figure 3. Datasheet hardware operation for ESP8266 ESP-12F MCU

The MCU will require ~80 mA of current and a range between 3.0-3.6V continuously for operation as shown in Figure 3, defined in the Power Subsystem how it will be delivered. This voltage will be set on pins 8 (Vcc) and 9 (GND). Ideally, we should be within the recommended 3.0-3.3V so minor fluctuations in the voltage do not go above 3.6V on the MCU. We will use GPIO pins 14 as the CLK signal, 12 as MISO, 13 as MOSI, and 15 as the SS to run SPI to communicate with the FPGA [2]. The local server, being hosted on a computer, will require power delivered through a commercial power adapter supply (i.e., laptop being powered by a laptop charger), however, we allow this to be hidden from view for the user.

The application server component allows the user (when connected) to upload an image through a computer or cell phone and enables the MCU to receive the data through a request following the event. The front end can be created with a simple interface (basic web development through HTML/CSS/JavaScript) that allows users to upload an image, and an on-screen button which initiates a POST request for the image to the server, which enables it as ready to be delivered to the MCU. The back end can be handled with the Django framework and a custom API which allows the user to upload the image on the server. After a successful upload, the MCU will receive an event through the WebSocket it is connected to (also on the server) to get an encoded version of the to-be-printed image (an efficient method being through base64 string encoding in a JSON) from the server.

As mentioned previously, the server can be hosted locally for the scope of this project as-is, especially as a means of saving a consistent amount of money as opposed to hosting on a commercial cloud platform such as AWS or GCP. For large scale implementation, we of course cannot rely on local servers, but this simplifies our testing requirements with a small sample set of users and devices to work with.

2.3 Imaging Subsystem

2.3.1 Description

The Imaging Subsystem allows for three pixels to be converted and mapped into the dithered equivalent after being processed. The processing is done entirely by hardware as this is the "hardware accelerator" portion of our project. This will allow for the images to be printed out at an incredible rate in a very similar fashion to how it is done in industry with consumer grade printers at HP. Additionally, this subsystem includes the thermal printer itself, which takes the hardware accelerator's image and routes it back through the board so that the MCU can send the image to the printer to be printed.

2.3.2 Requirements and Verification

Table 2: Imaging Subsystem R&V

Requirement	Verification
• Processing an image with the FPGA should be faster than processing it with the microcontroller.	• We will run a timing test between the microcontroller and the FPGA. By doing this, we will be able to clock the time between the time it takes to process the image using the FPGA and the amount of time it takes to process the image using the microcontroller. The FPGA's time should be relatively faster than the MCU's time with software implementation of the image processing to demonstrate hardware acceleration. o This is done by using the C libraries and calling clock() and having an equation to calculate the amount of time a certain function takes on our C model. o Then we will be able to figure out the time on the FPGA by multiplying the inverse clock (50 MHz) by the amount of clock cycles (3) by the amount of pixels in the height and the width of the picture.
• The FPGA should be able to take in picture data as 8-bit pixels and output it as a 1-bit pixel map. o The FPGA needs to be able to have proper handshaking logic with the microcontroller.	• We will run a testbench on the FPGA to see if the data is able to be processed correctly. To do this, simply load a small 16*16 image and individually manually check each pixel to see if the data is being processed correctly by following the algorithm that the FPGA is following. o To check if the handshaking is correctly being done between the microcontroller and the FPGA, we should monitor signals such as transmit ready and transmit receive

	correctly between the two modules. Then, we can test if the ready and receive signals update when we try to communicate between the two.
● The FPGA should be able to function by itself as soon as it is turned on, without having to be flashed again by an external source – simulating a commercial printer.	● Upon the plugging in of all the components, we should check to see if the original FPGA ROM is loaded into the FPGA. If it is not (the LEDs are not blinking rapidly), then we would know that the FPGA has successfully loaded in our correct .sof file.
● The temperature of the surface of the enclosed system (with the imaging subsystem) must not exceed 120 degrees Fahrenheit.	● The temperature of the printer system as a whole is important for user safety, in which it cannot exceed a recommended temperature limit of 120 degrees F [9] for handling. ● During the startup, idle, and active printing stages of operation we will measure the temperature of many surfaces of the printer enclosure using an infrared thermometer. ● We will record the temperatures of every tested region in Fahrenheit, ensuring it does not exceed 120 degrees F, as this enclosure is for our final design. ● If it does exceed this temperature at a certain position, we will need to acknowledge that position on the enclosure is not safe for direct contact.

A DE10-Lite FPGA will be utilized to simulate the operation of an ASIC which is not openly available to the mass public. FPGAs are commonly used to test HDL code at a very cheap cost compared to a full-scale tape out, albeit at a slower clock, so we will attempt to multiply our hardware throughput at the correct proportional speedup rate. The FPGA can run at a speed of 50 MHz [7], while mainstream ASICs can usually run 10-50x faster than this, but this will still be much faster than processing the image through software means (on the cloud or on the MCU).

The FPGA must take in data through the SPI protocol and be able to send data back out through the SPI protocol as well. The FPGA will take in three pixels and run it through a pipeline. Firstly, the FPGA must store all the RGB (red, green, and blue) values of an image into its onboard memory to prepare it to be processed. While the pixels are being stored into memory, we can start processing some of the data while it is still in the process of gathering data from the MCU. This is because many of the algorithms that will be applied, such as Floyd-Steinberg dithering [6], only require 5 adjacent pixels for the image to start being

processed. We need to set up a state machine that detects whenever a threshold number of pixels have been loaded into the FPGA, and then, it will start to process this data simultaneously. The third stage of the pipeline is when the data needs to be stored in a final bitmapped processed stage, and then this final image will be sent back out into the MCU and will be ready for printing. This process happens very fast, and doing the math, it should not take more than 3 (Number of pipeline stages) * (xy) = 3xy clock cycles to process a single image, where x and y are the dimensions of the picture.

Floyd-Steinberg Dithering

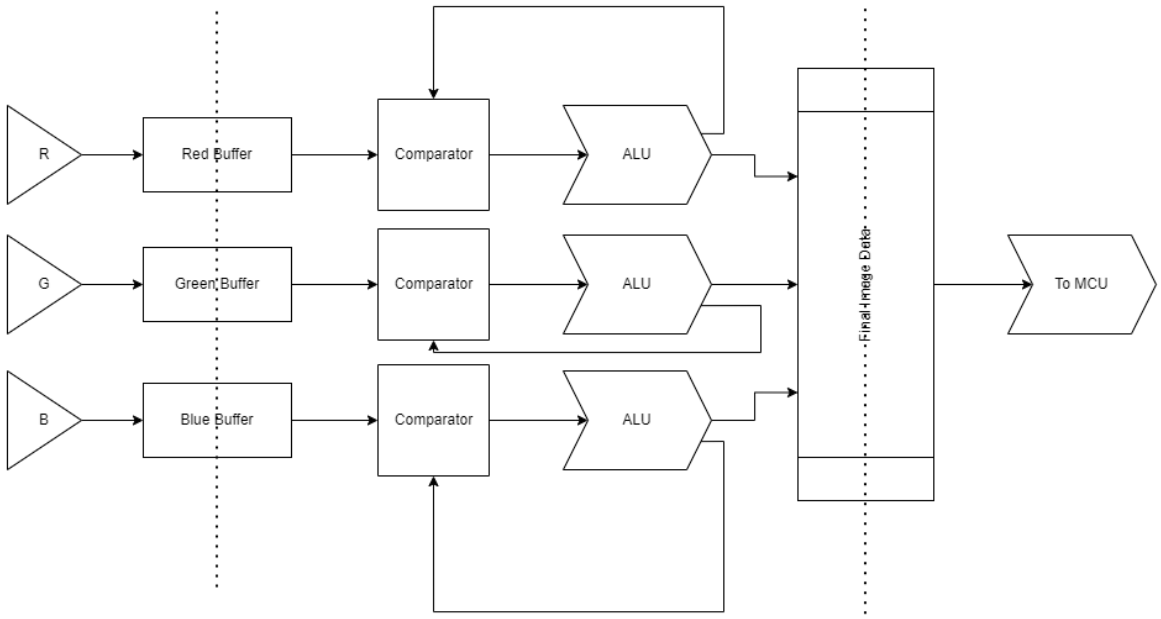

Figure 4. Floyd-Steinberg Dithering block diagram.

As for the printer, the printer must be able to interact with the MCU correctly. This means that the ports coming out of the MCU have to be connected correctly to the printer. The firmware that is needed for the printer to print is provided already by Adafruit through its proprietary libraries, however we need to figure out for ourselves how the connection between the printer and the MCU is done as well. The connection between the printer and the MCU is TTL serial, and it can be done over 3.3V or 5V so there is no need for any level shifters. However, we also need to make sure that the printer receives enough power so it will be run with power through its own dedicated rail since we expect that at least 10W will be used by the printer during peak run time.

2.4 Board Subsystem

2.4.1 Description

The Board Subsystem is an interactive and diagnostic block that allows for the user to check the status of the entire system briefly.

The primary component is a small 1.8" raw TFT display **[4]** that displays useful information about the battery level and the status of a printing job for user diagnostics (e.g., completed, failed), all of which is processed and delivered from the MCU.

There will also be an infrared receiver sensor that will sense if there is still a supply of thermal paper for the thermal printer to print on. If the sensor detects a change in the paper supply, this information will be sent to the MCU, which will have the LCD print out a visual warning/error and prevent the printing process.

Finally, there will be a switch box that the user can use. Switches, when turned on and off, will change which algorithm the FPGA will use when processing the image to “fine tune” it better for various printing applications (e.g., Floyd-Steinberg Dithering, Burke's Dithering **[5],** etc.).

2.4.2 Requirements and Verification

Table 3: Board Subsystem R&V

Requirement	Verification
• The diagnostic LCD must be responsive if the system status changes.	• Start at a known system state where the diagnostic LCD is outputting a steady state. • Change the system state by either hardcoding a state change or disabling the printer. At this point, start recording time elapsed on a stopwatch. • Stop the stopwatch when the diagnostic LCD updates its display to the correct status. • Record the start-stop readings and the time taken. Repeat for all possible system status changes. • The ideal time for all changes should be < 5 seconds guaranteed, reflecting upon the system status with little lag for the user.

● The “algorithms” switchbox controls the image processing algorithm that the FPGA uses.	● Start with all switches in the OFF position. Change one of the switches to ON. Record what algorithm this switch should map to. ● We will write a TestBench file that allows us to analyze the value per bit after the FPGA processes the image. ● Since the image processing algorithms are deterministic, we can compare values at specific subarrays. We will check that values are correct through a Python script that performs the naïve dithering algorithms (guaranteed to work) and asserts all the values are the same.
● At startup, the diagnostic LCD status should read “Ready” before printing.	● Check that there is enough paper in the printer beforehand, and that the system is idle (i.e., just starting up) ● Power the system on, with the printer plugged into a power supply running at 5–9V and 2A, power ground to the power supply ground, the printer ground to the microcontroller ground. ● Connect the TX pin of the microcontroller to the RX pin of the thermal printer ● We will not be initiating any transfer of data for this requirement test, and instead the microcontroller should be programmed to display “Ready” on the LCD once connected to the server WebSocket. ● We will manually verify that the LCD reads “Ready” at this stage of operation, demonstrating that the microcontroller is connected properly and ready to listen for events from the server.
● The diagnostic LCD must display correct information (considering delay) about the battery. The value of the	● Ensure that the system is powered off completely initially. ● At time of powering the system on, we can force the statuses of different situations to occur.

battery level must be within 10% of the actual battery level.	● At various charge levels of the battery, a circuit with different node voltages in order to indicate the current battery level of the battery pack using LEDs that have different forward voltages. We will use an oscilloscope to measure the voltage across each node to ensure they measure the right remaining voltage in the battery, and record the results.
● While the system is on, start a printing job with paper loaded. The status displayed should be “Completed” when the thermal paper finishes printing.	● Check that there is an adequate amount of paper in the printer prior to testing ● Connect the printer power to a power supply running at 5-9 V and 2 A, power ground to the power supply ground, the printer ground to the microcontroller ground. ● With the TX pin of the microcontroller connected to the RX pin of the thermal printer, we can feed in serialized sample data to print (i.e., a hardcoded bitmap) ● Upon completion of the printing job, we will manually verify if the LCD displays the word “Completed”. Otherwise, this indicates there is an issue with the microcontroller’s ability to identify the end of the program.
● While the system is on, start a printing job with no paper loaded. Check the LCD, and the status displayed should be “Failed”, not “Completed”.	● Check that there is no thermal paper loaded in the printer prior to testing ● Connect the printer power to a power supply running at 5-9 V and 2 A, power ground to the power supply ground, the printer ground to the microcontroller ground. ● With the TX pin of the microcontroller connected to the RX pin of the thermal printer, we can attempt to feed in serialized sample data to print (i.e., a hardcoded bitmap) ● The IR sensor should recognize this, send data to the microcontroller which will refuse the print job. We will

	manually verify that the word “Failed” is shown on the LCD.

This block contributes to the overall design by providing a reasonable level of user experience. It informs the user of potential issues pertaining to battery life and printer status and allows the user to change between different image processing algorithms based on their printing needs. While not directly responsible for the Board Subsystem, the MCU is responsible for sending and processing data that is delivered to this subsystem. Without it, the LCD would fail to function, the sensor data would be unable to be processed, and as a result, the Board Subsystem would essentially be rendered useless.

The board subsystem has the components of all the sensors, the MCU, and the LCD. These components all together all draw 3.3 V and each <500mA of current collectively. We shall be able to power this subsystem quite easily on its own line since the amount of power drawn is quite small compared to the imaging subsystem.

2.5 Power Subsystem

2.5.1 Description

The power subsystem supplies power to every other subsystem. Namely, it powers components such as the ESP8266 ESP-12F MCU at 3-3.3 V [2], the thermal printer at 5-9 V [3], the FPGA at 5 V [7], the LCD at 3.3 V [4], and the infrared sensor at 3-5 V [10]. Its components are a USB-C controller that will be connected to a PC's USB-C port. This connection will supply and charge power to a battery pack that is protected by its own integrated circuit. There will also be a voltage regulator system in this subsystem that will step voltage input levels to the correct output levels per component. It will also flash the MCU (send program information to the MCU to execute).

This subsystem is necessary for supporting the continued operations of the entire system, which includes displaying the diagnostic information, the Wireless Subsystem receiving image data, processing image data, and printing.

2.5.2 Requirements and Verification

Table 4: Power Subsystem R&V

Requirement	Verification
● The power provided this subsystem generates must be sufficient for the entire system.	● We will measure the voltages across other components with a multimeter (insert the leads into the pin and ground). The voltage ranges and current ranges accepted for verification are: o ESP8266 ESP-12F MCU 3-3.3 V @ 80 mA [2] o Thermal printer 5 - 9 V @ 2 A [3] o FPGA 5 V ± 5% @ 0.5 A [7] o LCD 3.3 V ± 5% @ 150 mA [4] o Infrared sensor 3-5 V @ 3 mA [10]
● The subsystem is safe. It does not exceed the hard limit of 20 W.	● While the system is active, ensure that there is no power overage by measuring the power. It should not exceed our cap of 20 W.

● The subsystem can push out code/flash to the microcontroller to program it.	● While the system is active, try to send C/C++ code to the microcontroller that tries to turn its LED on and off. ● If the test LED turns on and off, we conclude we have successfully flashed the MCU with our test code.
● The subsystem must be able to provide power to the entire system for 1.5 hours from full charge without recharging during this time.	● Verify that the batteries are fully charged by measuring the voltage drop across it with a multimeter or oscilloscope. This voltage drop should be around 7.4 V. If not, then charge the batteries until this condition is true. ● Plug the PC in to the USB-C controller and turn the system active. ● Start a stopwatch as a user starts the first printing job since the system turned on by uploading an image to the local server. ● Have the user constantly upload images as the printing jobs are completed. ● If the stopwatch records 1.5 hours and the system is still active, this requirement has been verified. Otherwise, this requirement has failed.

2.6 Tolerance Analysis

2.6.1 Power

The Power Subsystem is the block in our system that is critical to the success of our project and the most challenging to us. There are two major requirements for the Power Subsystem that relates to tolerance: it must generate enough power to all components without exceeding the required limits, and it must not exceed a limit of 20 W. This is a generous limit given that we have the power requirements above to be:

$$P_{system} \leq P_{MCU} + P_{Printer} + P_{FPGA} + P_{LCD} + P_{IR} \quad (1)$$

$$P_{system} \leq 3.3\,V * 0.08\,A + 7.4\,V * 2\,A + 5.025\,V * 0.5\,A + 3.465\,V * 0.15\,A + 5\,V * 0.003\,A$$

$$P_{system} \leq 17.87365\,W$$

To achieve the two requirements above, we use a voltage regulator circuit so that our components don't have excessive voltage drop. If this were to be the case, our components may overheat or burn out.

Since we want our system to last at least 1.5 hours on battery power, we must choose a set of battery packs that satisfy:

We are drawing an estimated current from all sources as follows:

MCU + FPGA + LCD + IR + Printer = Total current draw (Peak Current Draw during printing)

$$0.08\,A + 0.5\,A + 0.15\,A + 0.003\,A + 2.0\,A = 2.733\,A \quad (2)$$

$$2\ battery\ packs\ combined\ @\ 2000\ mAh\ each = 4000\ mAh$$

$$4000\ mAh\ /\ 2.733\ A = 1.465\ hours\ assuming\ peak\ current\ draw$$

2.6.2 Imaging

The printer itself needs to operate at over 150 degrees Fahrenheit to activate the thermal paper, and therefore we must ensure, for the safety of the device for the user, that the specific area intended to be held by the user remains under 120 degrees Fahrenheit throughout operation. The reason for 120 degrees Fahrenheit is because this is generally agreed upon for handheld products as the upper limit of a safe-to-touch temperature [9], and it would be extremely detrimental if the device were to cause harm by exceeding this rating.

2.6.3 Data Transfer / Wireless - Board - Imaging

We will try to compute a bad case printing job but not so bad that an HTTP request hangs for an indefinite amount of time. In that case, it will take an infinite amount of time to finish printing - it may not even finish the printing job even.

We assume that the worst case scenario is where our images take 5 seconds to upload to our local server, this is dependent on our internet speed and latency of our network. Now, the

MCU must send a GET request to retrieve the image data. The time taken to complete this request depends on the size of the request, time needed to encode the request data, bandwidth of the shared network, and even the server load. Because there are so many variables, let n be the time (in seconds) it takes for the MCU to complete the GET request and receive image data.

From the datasheet, data transfer with the RX and TX pins of the MCU can reach 4.5 Mbps (megabits). Assuming our image is an x by y image, where x = pixels per row of the image and y = pixels per column of the image and each pixel is represented by a byte, it will take (xy)/562500 seconds to output the image data to the FPGA.

We need the FPGA to process the image using its algorithm, which has a runtime of O(3xy) clock cycles.

Then, we have the final time needed to print the image on the thermal printer, which will take O(xy*c) time. This is because printers print out rows, and for a fixed row, they print from one side of the paper to the other side of the paper. The following computes for c.

Now, assume the printer has to print out a width of 48 mm per row, which is the print width on the thermal printer, (so x = 48a, where a is some constant) but is reasonably bounded by a height of h mm per row (so y = hb, where b is some constant). Assume that a dot is equivalent to a pixel for simplicity.

The thermal printer can print 60 mm/sec to 80 mm/sec (at 8.5 V) [11]. Let's assume we have a slow thermal printer such that it prints at a rate of 60 mm/sec. This number means that the thermal print head moves at 60 mm/sec as it prints at the head. Therefore, it can print out a row of 8 px tall in 0.8 seconds. For a given h, it will take 0.8 seconds * h seconds. Assuming that the print head needs to realign every row back to the other side of the paper, the head needs to move 48.0104 mm, which takes an additional 0.8002 seconds. Therefore, for a given h, it will take 0.8 * (h / 8) + 0.8002 * ((h / 8) - 1) seconds. For example, if y = 100 px, it will take around 12 seconds to print.

The total runtime is therefore

$$T(x,\ y) \leq 5 + n + \frac{xy}{562500} + (0.8\frac{8y}{b} + 0.8002(\frac{8y}{b} - 1))\ seconds \qquad (3)$$

3. Costs

3.1 Analysis

Table 5: Parts Cost

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)	Link
ESP8266 SMT Module - ESP-12F	Ai-Thinker	$6.95	$1.976/ea	$9.88 For 5x	Amazon
Thermal Printer	Adafruit	$49.95	N/A	$49.95 For 1x	Adafruit
USB A to Micro USB-B Cable	CableMatter	$4.99 For 3x	$1.663/ea	$4.99 For 3x	Amazon
ESP8266 SMT Module Breakout Boards	Songhe	$9.95	$2.776/ea	$13.88 For 5x	Amazon
FPGA DE-10 Lite	Intel	$179.99	N/A	$0	Amazon (Free from UIUC)
Lithium Ion Battery Pack	Urgenex	$24.99	N/A	$24.99	Amazon
IR Sensor	Adafruit	$2.95	N/A	$2.95	Adafruit
LCD	Adafruit	$9.95	N/A	$9.95	Adafruit
LEDs (x30) UHD111A-FKA-C3K23E1L3VG5ZB3Z3	Cree LED	$0.10	$0.034/ea	$1.02 For 30x	Mouser
Resistors 100 ohm (x100) RC0402FR-07150RL	Yageo	$0.10	$0.00580/ea	$0.58 For 100x	Digikey
Capacitors 0.1uF (x100) 0402YC104KAT2A	KYOCERA AVX	$0.10	$0.00640/ea	$0.66 For 100x	Digikey
USB IC Controller	STMicroelectronics	$3.49	N/A	$3.49 For 1x	DigiKey
USB C 3.1 Receiver	Tulead	$11.59	$0.2318/ea	$11.59 For 50x	Amazon
Total				**$145.98 (Adjusted for 9% sales tax)**	**…**

Assume the market rate for an ECE major graduate is 35$/hour. We assume we will work on the Portable Thermal Printer for 15 hours a week for the rest of the weeks, assumed to

be 9, in this semester, for which we will get paid for. Then, the labor cost for producing the Portable Thermal Printer will be:

$$C_{labor} = 35\ dollars/hour\ *\ 15\ hours/week\ *\ 9\ weeks\ /\ engineer\ *\ 3\ engineers\ =\ 14,175\ dollars \quad (4)$$

And to produce it, we need to purchase the above parts:

$$C_{parts} = \$145.98$$

$$C_{total} = C_{labor} + C_{parts} = \$14,175 + \$145.98 = \$14320.98$$

3.2 Schedule

Table 6: Work Schedule

Week	Gally	Jason	Kevin	Everyone
2/20	Power Schematics	Design Web Server	Design Web Server	Design Doc
2/27 (PCB Review) Finish Wireless Subsystem!!!	HAS EXAM FEB 27th HDL Verification	Verify functionality of web server	Verify functionality of web server	PCB Traces Layout Proposal v2
3/6 (First Round PCB)	Verifying functionality of thermal printer with ESP8266 **Prioritize Everyone Task**	HAS EXAM MARCH 9th Running C model of FPGA on ESP8266 **Prioritize Everyone Task**	HAS EXAM MARCH 9th Writing Firmware to Interface ESP w/ Printer **Prioritize Everyone Task**	SPI Interface between FPGA and MCU Order batteries & IC's for USB-C HDL Verification
3/13	Verify Functionality of Power Subsystem PCB	**Prioritize Everyone Task**	Verify Functionality of Power Subsystem PCB	SPI Interface between FPGA and MCU HDL Verification
3/20 Finish Imaging Subsystem (HDL)!!!	Spring Break	Spring Break	Spring Break	Fix Potential Issues in PCB and resubmit! Order Voltage Regulators
3/27 (Second Round PCB)	Integrate Jason & Kevin's Firmware	LCD Firmware	Sensors (IR & Battery Sensor) Firmware	Power Subsystem Whitebox Testing
4/3 Finish Board Subsystem!!!	Assemble Imaging System w/ Board System	HAS EXAM APRIL 3rd Verify Imaging System Works with Board System w/o batteries	HAS EXAM APRIL 4th	Begin full system integration
4/10 Finish Power Subsystem!!!	HAS EXAM APRIL 10th Verify Power Subsystem works!	Assemble Power Subsystem	Verify Power Subsystem works!	Find an enclosure for Portable Printer

4/17 (Mock Demo)	Buffer Week for final touch ups	Buffer Week for final touch ups	Buffer Week for final touch ups	
4/24 (Final Demo)	-	-	HAS EXAM APRIL 27th	Final Demo/ Final Papers / Finish lab notebook
5/1	-	-	-	Final Presentation / Final Papers / Finish lab notebook

Most of the scheduling that happens for this project happens on an individual subsystem basis. We plan to finish the most important parts of the subsystem first to achieve partial credit if there is not enough work done by the deadline. Therefore, we plan to implement and design the core parts of the project such as the microcontroller and the FPGA first. We should also plan to interface with the printer as well, so we have a system that could print out a bitmap first. After we finish this core system and demonstrate a successful print, we can then start working on the power and the rest of the peripherals of our system such as the switches and the LCD.

We shall split up the work so that Gally interfaces more with the FPGA while Jason and Kevin work on the microcontroller more. This is because Kevin and Jason both are computer engineers who have more experience with coding high level languages and being able to utilize the microcontroller better in its environment. This will prove useful as well since the microcontroller is the brain of the entire system and it controls how the rest of the system will behave. The FPGA on the other hand requires more experience with HDL, and Gally has more support from his connections at HP and his experience with ECE 411 for this portion. For the power electronics, we split it up very evenly due to our lack of experience with the USB C power and data delivery. We will need to work as a group on that portion of the project and we should consult outside resources who specialize in power such as the head TA and Professor Gruev for this.

4. Ethics and Safety

Our project aims to create a convenient and speedy printing solution for consumers. As such, we must ensure that the design of the printer system is safe for users to handle and interact with on a consistent basis. While many of the components listed are found in everyday objects and are consumer grade, there are some components that may pose risks in their usage. According to the IEEE Code of Ethics I.1 [1], we must remain transparent with our design process and disclose the factors that might endanger the public and/or environment.

One concern that is associated with our power subsystem is the use of battery packs. These batteries are lithium-ion rechargeable batteries, which, due to their energy density, makes them susceptible to explosion when in contact with high temperatures or manufacturing defects. This was demonstrated with the Samsung Note 7 smartphone case study [8], where the battery packs caught on fire due to a defect in the phone design that allowed the battery leads to touch and short circuit. We aim to follow all appropriate guidelines and documentation related to proper charging and discharging of the battery, ensuring it operates at a safe temperature and delivers the correct voltages and current to the rest of the system. As the most volatile part of our system, we need to monitor the battery's performance with the most scrutiny. We will follow the lab safety manual for batteries at https://courses.engr.illinois.edu/ece445/documents/GeneralBatterySafety.pdf for guidance in order to prevent the potential danger in overcharging and discharging the batteries.

Another concern that is associated with our system is the printing subsystem itself. We are using a thermal printer, as previously mentioned, which can reach upwards of 150 degrees Fahrenheit during the printing operation. For the common case, where a user will meet the printer while it is both in and not in use, we need to ensure that the printer is safe to handle. We intend to design an enclosure that can allow users to handle and transport the printer safely, while not compromising on the overall functionality. We will also thoroughly test and record the temperatures of the surface of the finished product in order to properly disclose necessary details about the printer's safety.

Through this project, not only do we expect to provide new innovation in the way portable printing is achieved, but we also expect to find new in-depth discoveries in image processing and power electronics. We will transparently discuss our findings in hopes of benefitting not only ourselves but to society and the engineering discipline as a whole.

5. References

[1] *IEEE Policies, Section 7 - Professional Activities (Part A - IEEE Policies*), IEEE Code of Ethics 2020.

[2] AI Thinker, "ESP-12F Datasheet," 2018. Accessed: Feb. 07, 2023. [Online]. Available at: https://docs.ai-thinker.com/_media/esp8266/docs/esp-12f_product_specification_en.pdf

[3] P. Burgess and Adafruit Industries, "Mini Thermal Receipt Printer," Nov. 2021. Accessed: Feb. 07, 2023. [Online]. Available at: https://www.mouser.com/datasheet/2/737/mini_thermal_receipt_printer-2488648.pdf

[4] Truly Semiconductors Co., "JD-T18003-T01." https://cdn-shop.adafruit.com/datasheets/JD-T1800.pdf (accessed Feb. 07, 2023).

[5] T. Helland, "Image Dithering: Eleven Algorithms and Source Code." https://tannerhelland.com/2012/12/28/dithering-eleven-algorithms-source-code.html (accessed Feb. 09, 2023).

[6] justinkraaijenbrink, "Exploiting the Floyd-Steinberg Algorithm for Image Dithering in R," Medium, Jan. 30, 2021. https://medium.com/analytics-vidhya/exploiting-the-floyd-steinberg-algorithm-for-image-dithering-in-r-c19c8008fc99 (accessed Feb. 09, 2023).

[7] Terasic, "DE10-Lite User Manual," Jun. 05, 2020.

[8] R. Rox, "Samsung Galaxy S7 explodes during charge," Notebookcheck, Sep. 03, 2017. https://www.notebookcheck.net/Samsung-Galaxy-S7-explodes-during-charge.246242.0.html (accessed Feb. 08, 2023).

[9] Johns Manville, "Too Hot to Handle?," www.jm.com, Feb. 25, 2015. https://www.jm.com/en/blog/2015/february/too-hot-to-handle/ (accessed Feb. 08, 2023).

[10] Vishay Semiconductors, "TSOP382.., TSOP384.., TSOP392.., TSOP394..," Feb. 2011. Accessed: Feb. 09, 2023. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/tsop382.pdf

[11] Adafruit, “A1 micro panel printer user manual.doc”, Accessed: Feb. 23, 2023. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/A2-user%20manual.pdf