Portable Thermal Printer

Gally Huang (ghuang23)
Jason Liu (jliu246)
Kevin An (kqan2)

TA: Hanyin Shao (hanyins2)

03/02/2023

ECE 445: Senior Design, Spring 2023

Team 29

Introduction

Our senior design project name is Portable Thermal Printer for ECE 445, Spring 2023.

We are Team 29, with the assistance of TA Shao Hanyin (hanyins2) and Professor Gruev.

- Gally Huang (ghuang23)
- Jason Liu (jliu246)
- Kevin An (kqan2)

Objectives and Background

Introduction

Problem

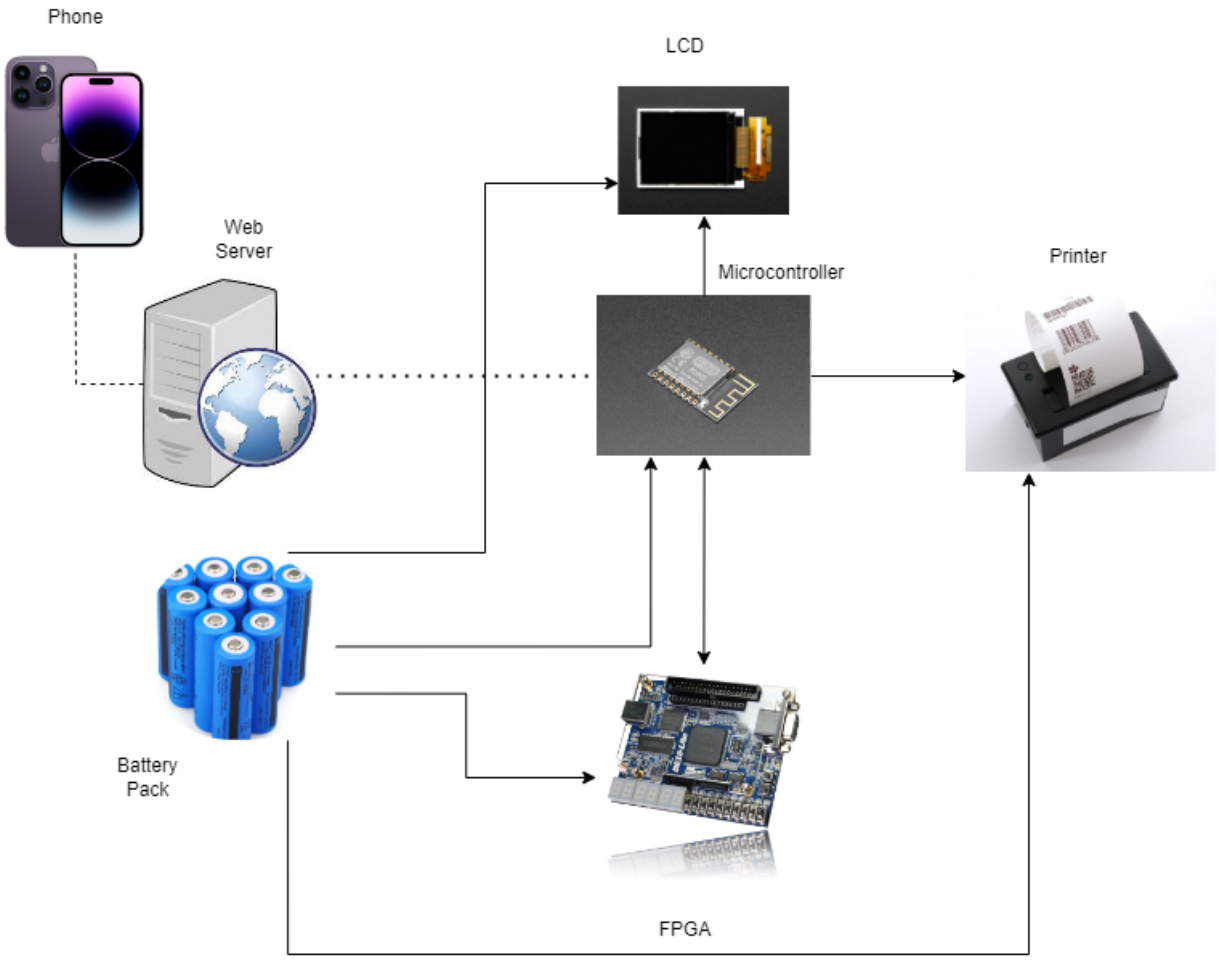
One of the biggest problems surrounding frequent travelers is the issue of portability. Items that are carried along have limits on their weight, cannot consume too much space, and have to compromise on quality. A target area that has been identified by the Hewlett-Packard Company (HP) lies within the commercial printer industry. Printers as a whole 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.

Solution

Our solution is a portable thermal printer, a system that receives wireless instructions for printing on receipt 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 (an FPGA) to implement our solution because they can stand in place for a real-world application-specific integrated circuit (ASIC) and eventually be developed 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 microcontroller unit (MCU) to handle the wireless components. Finally, we will be creating our own input/output shield (I/O shield) for the PCB that has the subsystems listed further down on top of it.

Visual Aid



Goals and Benefits

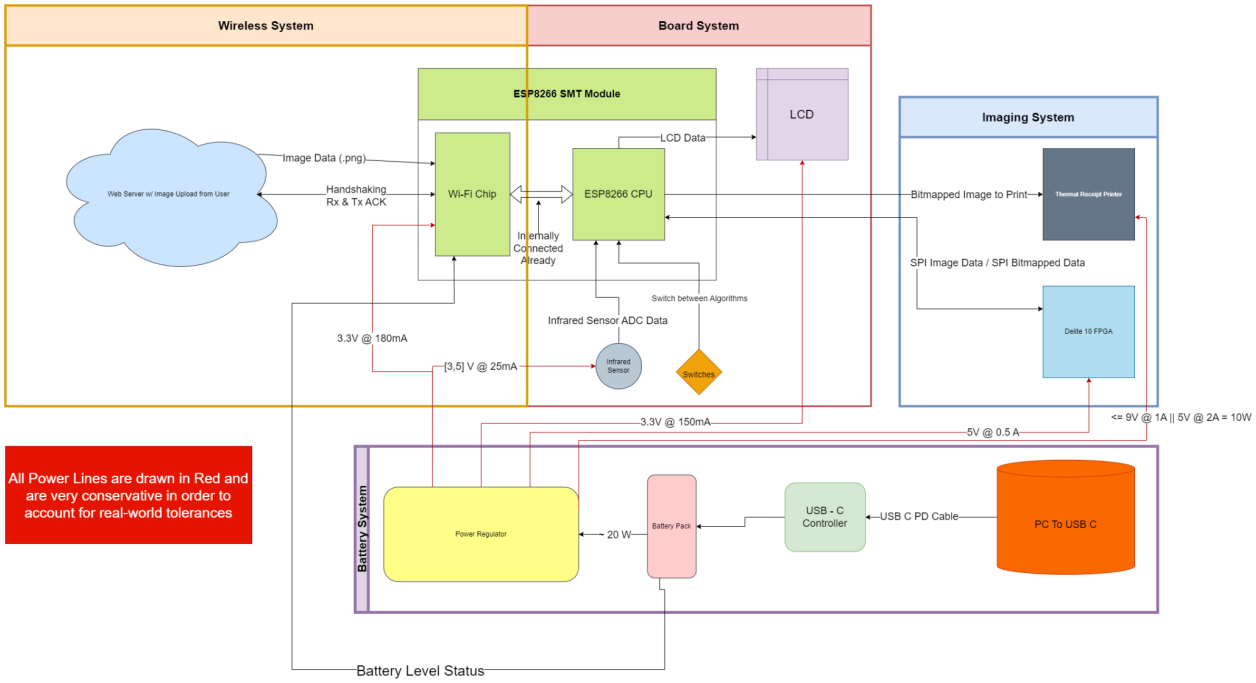
- Make printing portable in the world where many other technology have already evolved to become more portable
- Current solutions using wireless connection usually support Bluetooth, which is short-range, or are expensive.
- The project allows for printing with a battery system and wireless uploading of image data, making it very portable.

High Level Requirements

- The device design is portable. It should be able to wirelessly and accurately get the user-uploaded image data from a server to the embedded MCU. It should sit as a small footprint of at most 12"x12" as to fit comfortably within a suitcase, allowing for ease of transportation.
- 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.
- The start to end time, between user upload and completing the printing, should be within 20 seconds so as to not consume too much time for the user.

Design

Block Diagram



Description

Wireless Subsystem

Wireless Subsystem Overview

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 WebSocket connection between the MCU and the server. After receiving this data, the MCU will further process and deliver it to the other respective subsystems.

Wireless Subsystem Requirements

ESP8266 ESP-12F Microcontroller (MCU):

- 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 WiFi 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 asynchronous request events 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) in order 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.

Application server:

- The application server 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 WebSocket event following the event. The front end can be created with a simple interface (i.e., 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 be sent to the server. The

back end can be handled with the Flask framework and a custom API which allows the user to actually upload the image on the server. This process should take a maximum of 5 seconds from initial time of upload request to appearing in the server storage. After a successful POST, the WebSocket event that the MCU will receive will contain the encoded to-be-printed image data (an efficient method being through base64 string encoding through 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.

- The MCU will require ~80 mA of current and a range between 3.0-3.6V continuously for operation, defined in the Power Subsystem how it will be delivered. This voltage will be set on the MCU 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 protocol 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.

Imaging Subsystem

Imaging Subsystem Overview

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.

Imaging Processing Subsystem Requirements

DE10-Lite FPGA:

- An 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).

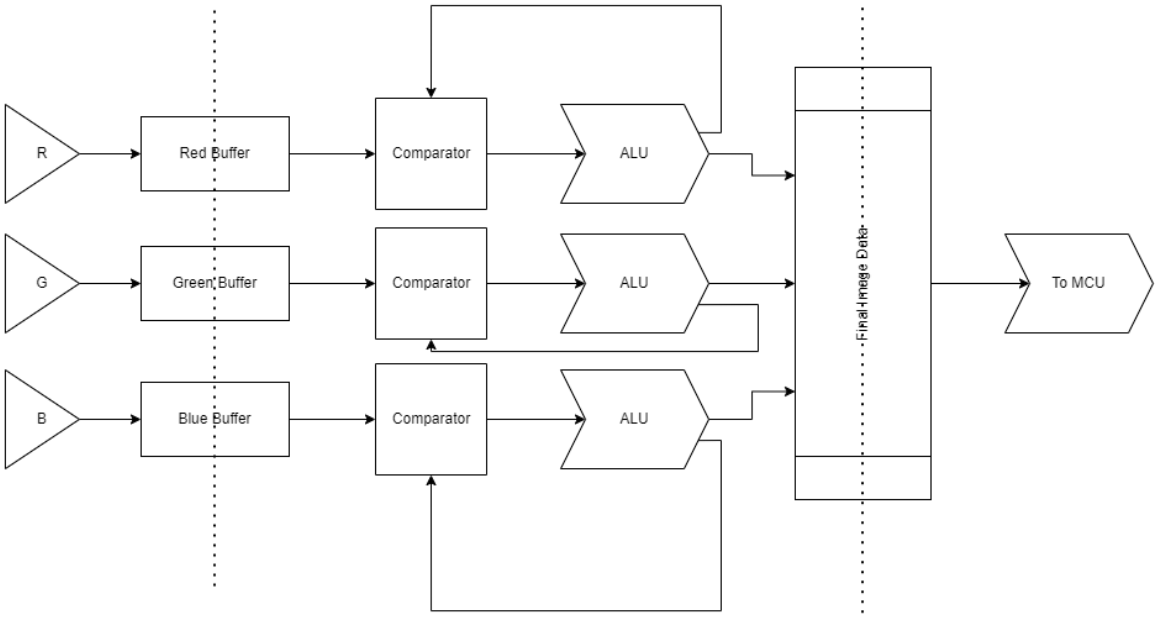
- The FPGA must take in data through the SPI protocol and be able to send data back out through the SPI protocol as well.
- The FPGA will take in three pixels and run it through a pipeline. Firstly, the FPGA must store all of the RGB (red, green, and blue) values of an image into its onboard memory to 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 requires 5 adjacent pixels for the image to start being processed. We need to set up a state machine that detects whenever a threshold amount 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 in order to process a single image, where x and y are the dimensions of the picture.

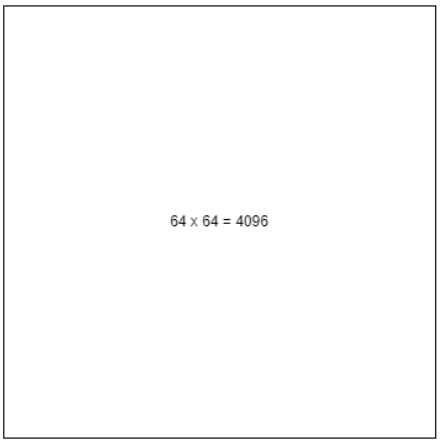
- The FPGA requires constant voltage of 5V and we can assume that it requires 0.5As as well since this is being outputted by the USB-A cable everytime.

- The printer requires constant power of 10W when it is active, whether this is through 5V or 9V is dependent on the input voltage to the system.

Diagram of sample algorithm (all are pretty similar except for different ALUs):

Floyd-Steinberg Dithering





- 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. 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.

Board Subsystem

Board Subsystem Overview

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

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, paper jam), 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 (e.g., Floyd-Steinberg Dithering, Burke's Dithering [5], etc.).

Board Subsystem Requirements

- 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 needs.
- One requirement for the LCD is that it must be able to refresh its status/display at a decent rate so that monitoring/debugging the system is reasonably convenient for the user (< 5 seconds). If something changes in the status of the system, the LCD should be able to reflect upon this change with little lag.
- 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 and as a result, the Board Subsystem would essentially be rendered useless.

Power Subsystem

Power Subsystem Overview

The power subsystem supplies power to every other subsystem. Namely, it powers components such as the ESP8266 MCU at 3-3.3V [2], the thermal printer at 5-9 V [3], the FPGA at 5 V [7], the LCD at 3.3V [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 power to our four 18650 batteries. We use a regulator system to

maintain constant voltage levels to the components stated above. It will also flash the MCU (send program information to the MCU to execute).

This subsystem as a whole 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.

Power Subsystem Requirements

- The other subsystems must be powered on with this subsystem at the stated voltage and current levels or with a maximum of -5% deviation.
- It is important that the power system is able to supply the upper conservative limit of 20W as well, since this would be able to provide enough power to the system in the case of sub components requiring peak power.
- We also must be able to check the current battery level percent of the 18650 batteries on the LCD in the Board Subsystem. This diagnostic data is to be delivered to the Board Subsystem for displaying to the user.

Risk / Tolerance Analysis

Servers are considered outside the scope of this class, so it may be difficult to implement. Additionally, based on our implementation of accepting data from a user, we can have our local server (and hence, our local device) be susceptible to a cyber attack. Using JavaScript makes us potentially vulnerable to some control flow hijacking, which can allow users to attack our device. Since our code is online, attackers can try to precisely send images to hijack the server.

The printer itself needs to operate at over 150 degrees Fahrenheit in order 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.

We will try to compute a bad case printing job but not so bad that a 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 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 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 $\frac{xy}{562500} seconds$ to output the image data to the FPGA. We need the FPGA to process the image using its algorithm, which has a runtime of $O(3xy)\ clock\ cycles$. The FPGA runs at a clock speed of 50 MHz, so we expect the runtime of image processing to be around $3xy\ cycles\ *\ (\frac{1\ second}{50*10^6\ cycles}) = \frac{3xy}{50*10^6}\ seconds.$

Then, we have the final time needed to print the image on the thermal printer, which will take $O(y\ *\ c)\ time$. This is because printers print out row by row. In particular, the thermal printer device we have prints out rows of 60 mm/sec to 80 mm/sec (at 8.5 V) [11]. The following computes for c. Assume that 1 pixel is equal to d dots.

Let's assume we have a slow thermal printer such that it prints at the minimum rate of 60 mm/sec. The printhead resolution is equal to 8 dots/mm, which means that per square millimeter of paper, there will be 8 ink pigment units. The greater the resolution, the sharper the image will be. Per second, the printer can print $\frac{60\ mm}{1\ second} = \frac{60\ mm}{1\ second} * \frac{8\ dots}{1\ mm} = \frac{480\ dots}{1\ second} = \frac{480\ dots}{1\ second} * \frac{1\ pixel}{d\ dots} = \frac{480\ pixels}{d\ seconds}.$

Therefore, for a given y, it will take the printer $y\ pixels\ *\ \frac{d\ seconds}{480\ pixels} = \frac{yd}{480}\ seconds.$

The total runtime is therefore…

$$T(x,\ y) = 5 + n + 2\frac{xy}{562500} + \frac{3xy}{50*10^6} + \frac{yd}{480}\ seconds.$$

- 5 -> Assumption based on worst-case requirement for image to take 5 seconds to upload to server
- n -> Time taken for MCU to fetch image data
- $\frac{xy}{562500}$ -> Time taken for MCU to transfer image data to FPGA plus Time taken to transfer finished image to printer
- $\frac{3xy}{50*10^6}$ -> Time taken for FPGA to process image data with algorithm
- $\frac{yd}{480}$ -> Time taken for thermal printer to print image

For example, if we have an image of 640 pixels by 480 pixels and d = 1, n = 1, it will take around 8.11 seconds to finish the printing job.

For the same image, if d = 3, it will take 9.1107 + n seconds, meaning as long as our MCU can fetch the image data in less than 20 - 9.1107 = 10.8893 seconds, the printing job will be finished in less than 20 seconds.

We believe that power is also a big issue for our printer. This is because there are many voltages running in parallel for the internals of the printer so we need to make sure whatever we are doing is safe not only for ourselves but for people around us as well. There should be voltage regulators between the battery and the entire system, even for devices that run on the same voltage such as the printer and the battery itself (7.4V). Nevertheless, other devices should also have a voltage regulator hooked up between it and the battery. We considered using LDO's instead of a switching buck converter for the rest of the circuit but this in turn is cause for concern since LDO's have a limited voltage dropout value that we cannot control. If the voltages are too far apart such as from the 7.4V to the 3.3V rail, then this would cause for the LDO to heat up too much and start a fire on the PCB. Therefore we decided that it would be best to layer our power system to be:

Battery -> 2x buck converters -> 3x LDO out of 2 buck converters.

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.

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