SnapLog Camera

Tianshu Wei

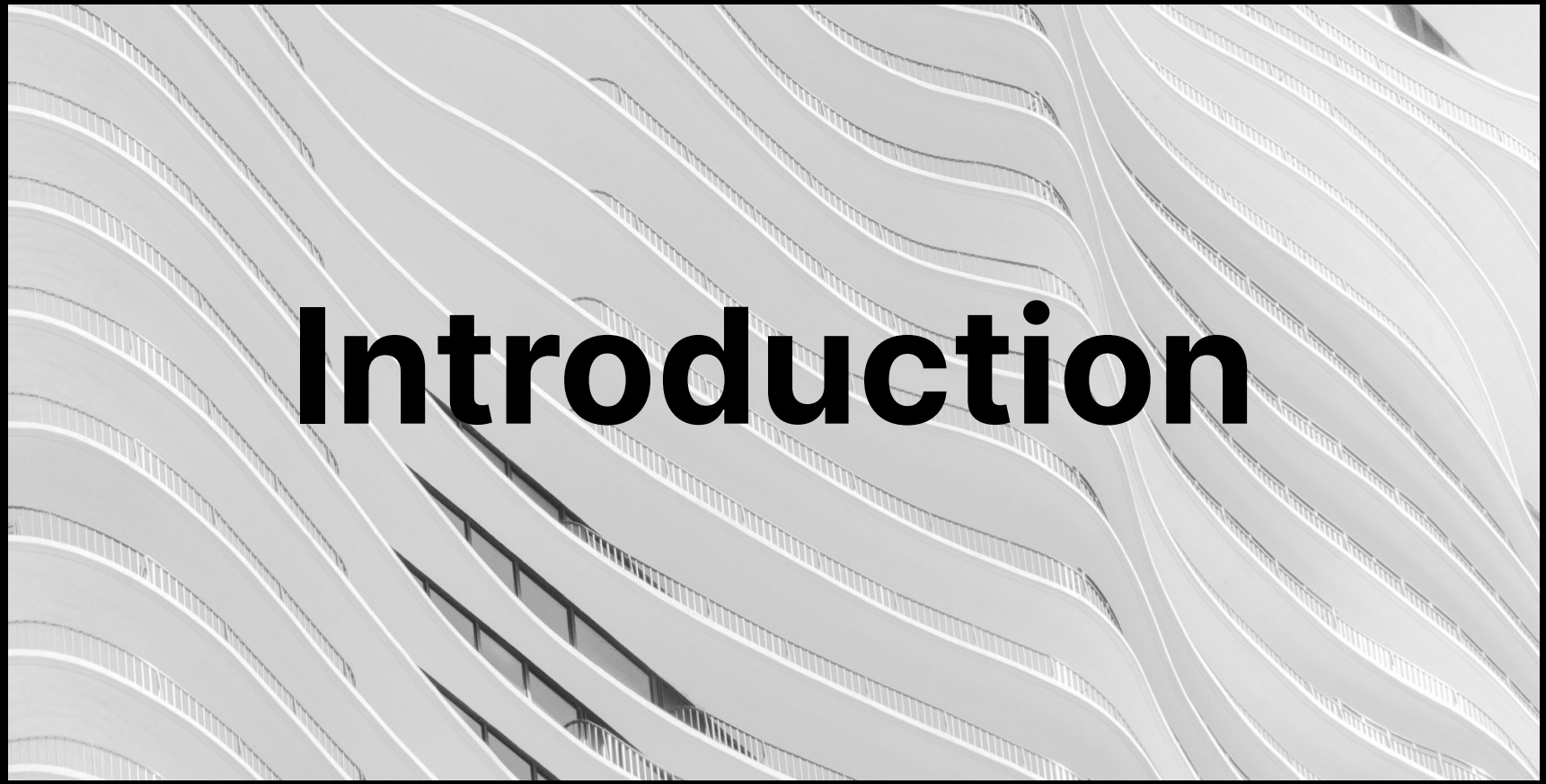
Fei He

Shuai Huang

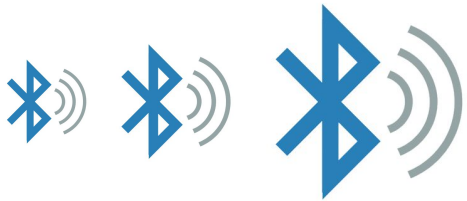

Objective

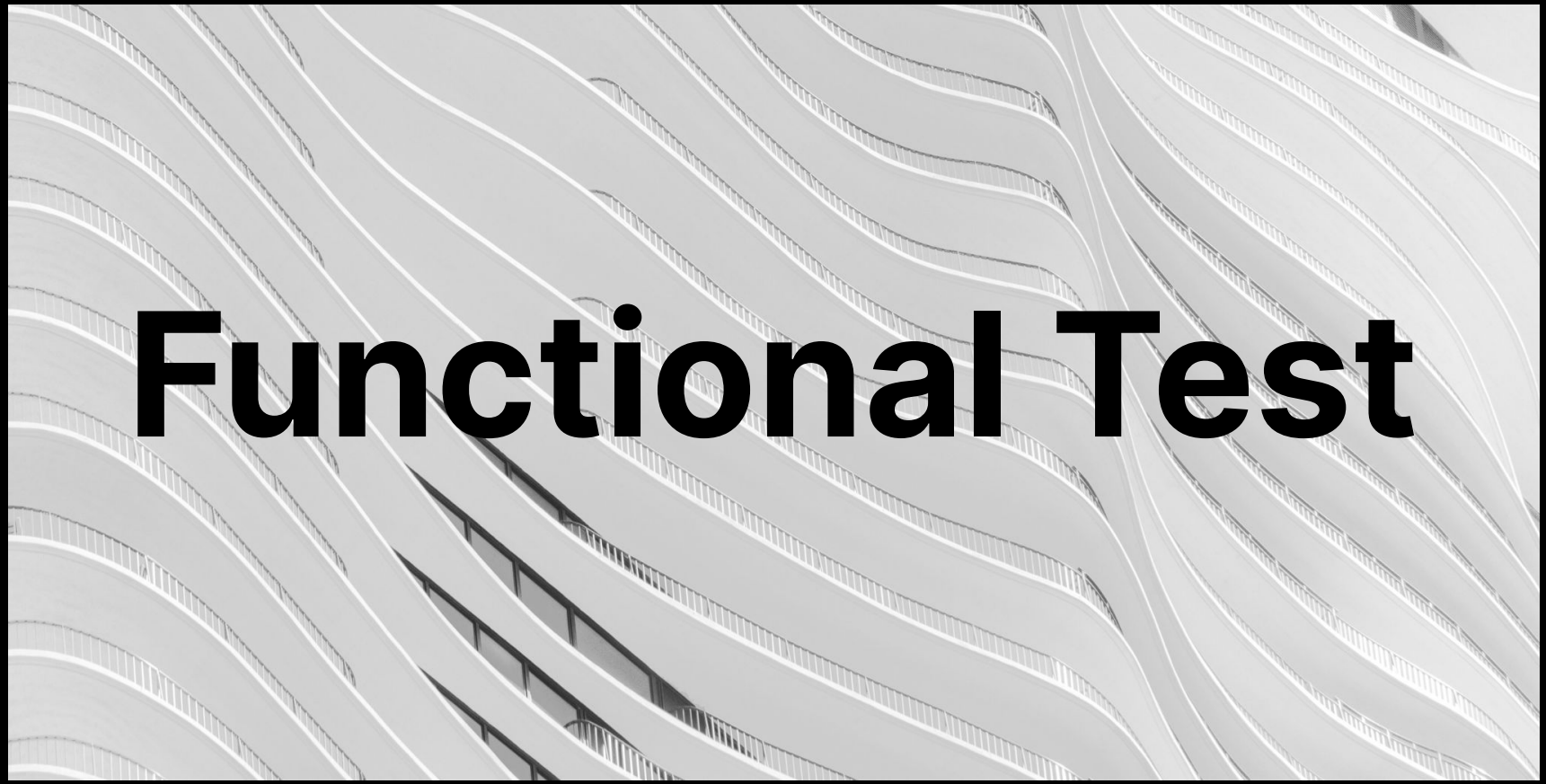
SnapLog Camera

Journaling can be repetitive and stressful for children who are asked to record their daily activities for schoolwork, as well as adults who may follow their habits for personal or professional reasons. Not only does it take time, but it also requires mental work to recall, note down, and often describe the events of the day coherently.


Introduction

Our project, SnapLog Camera, is a camera necklace that automatically takes photos and compiles into a captivating timelapse video. SnapLog generates a timelapse—a series of images shot at regular intervals that, when combined, provide a dynamic overview of your day. The camera can be set to take an image every few minutes, and each one is quickly and securely delivered to your smartphone.




Functional Test

High-Level Requirement

1. The camera is able to take pictures and send them to the host device. The camera will take a photo and transmit it wirelessly, and automatically, to another device. The delay should be within 5 minutes.

To surpass this core requirement. The team utilized BLE for efficient transmission, on top of which, we designed a reliable data transmission protocol, ensuring robust and reliable data transfer. The microprocessor was able to quickly wake up from hibernation and communicate with sensor and transmit parameters over SCCB everytime reliably. Remarkably, we were able to achieve battery life of an impressive theoretical lifespan of 533 hours, equating to an extraordinary 22 days of continuous operation, using only a tiny 800mah battery.

2. The camera should be able to take pictures automatically within a certain configurable interval, maximum 5 minutes.

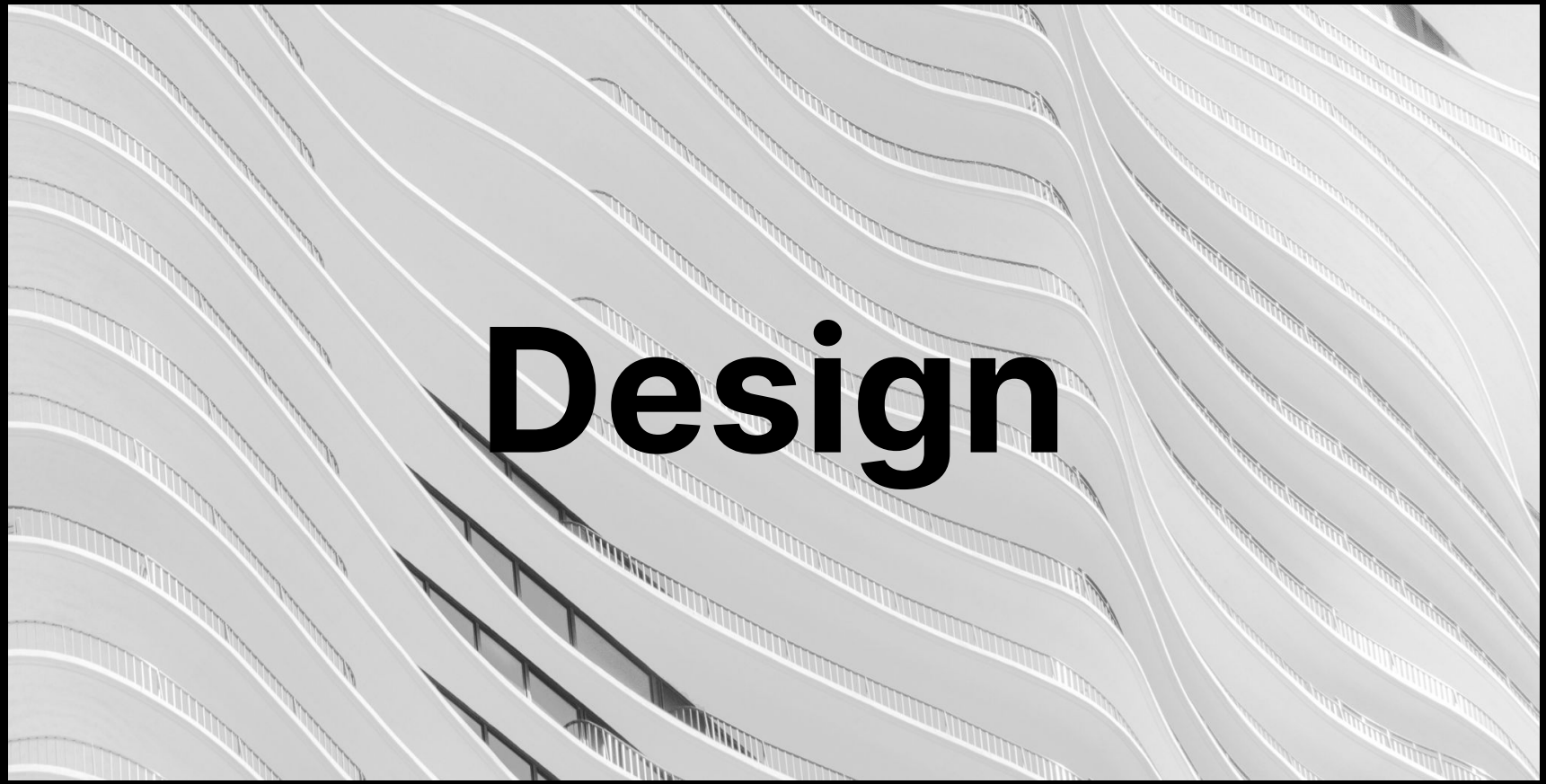
To complement this feature, the team integrated an intuitive GUI, facilitating effortless interaction with the hardware.

3. Finally, the software on our host device should be able to convert pictures taken in the last 24 hours into video that could be viewed on the device.

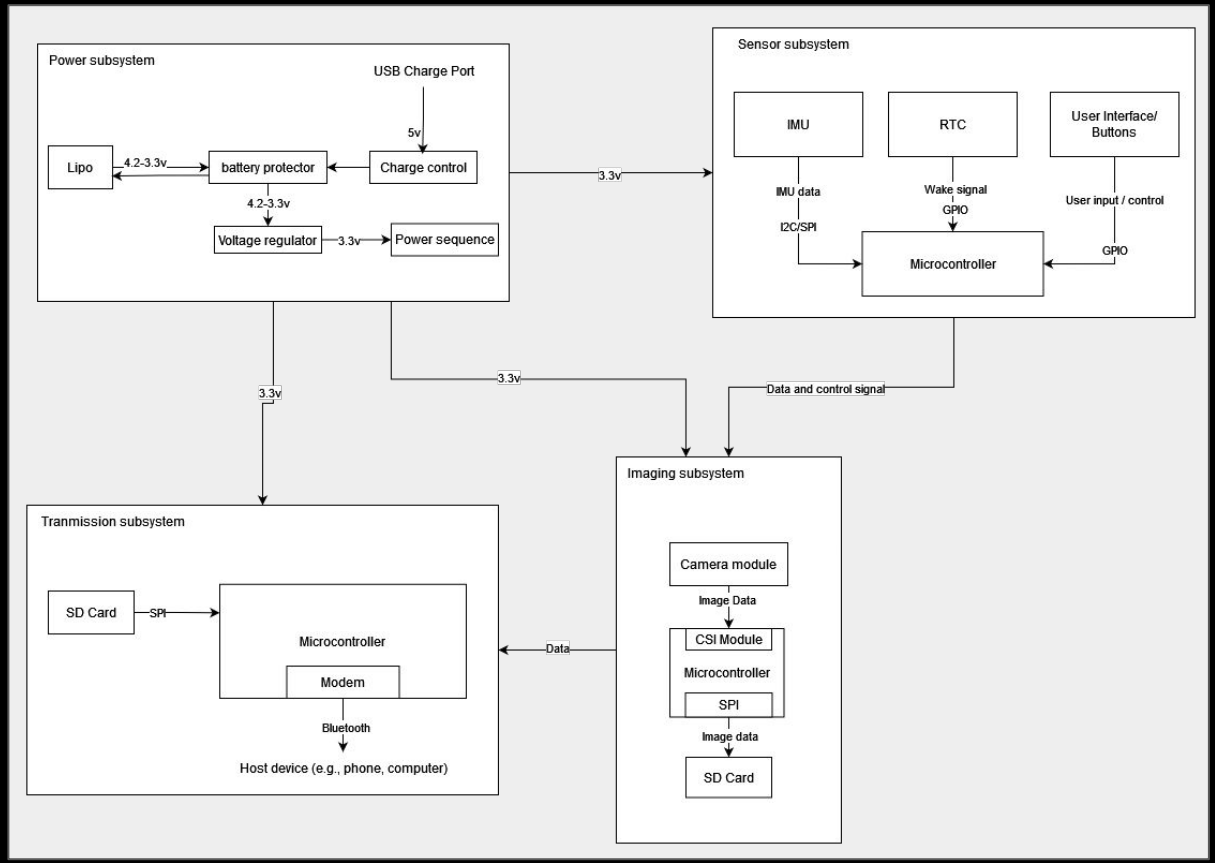
With these functionalities seamlessly integrated, the project delivers on its promise of simplifying the process of documenting and enjoying daily moments, making it a convenient and enjoyable tool for users seeking to preserve and cherish their memories.

RV Table

Requirement	Verification
The camera must be able to take pictures and send them to the host device within 5 minutes.	Test the camera's image capture and transmission time and verify that the images are received by the device within the specified time.
The camera should automatically take pictures at a configurable interval not exceeding 5 minutes.	Set different intervals (1 min, 3 min, 5 min) and ensure the camera takes and sends pictures as configured without user intervention.
The software on the host device must convert pictures taken in the last 24 hours into a video that can be viewed on the device.	Use the software to process images taken over 24 hours and check if it successfully creates a video.
Power subsystem should supply at least 3.3v 200mA to other subsystems.	Measure voltage and current over the resistor connected to the output of the power subsystem.
Imaging subsystem should generate raw imaging data that can be read by the microcontroller.	Verify the data generated by the imaging subsystem is readable and correct.
Transmission subsystem should wirelessly transmit images to other devices using Bluetooth.	Check the receiving program on the host device to ensure it correctly displays the images sent from the camera via Bluetooth.
The camera must have an immediate picture-taking function triggered manually by the user.	Test this function by manually triggering the camera to take a picture using a button. Verify that the picture is taken and processed immediately.
The camera must include a manual power-off function that can be activated by the user	Test the power-off functionality by activating the power-off button. Verify that the camera powers down immediately without any issues.


Design

Subsystems Design



Component selection

There are a few components are critical to our project so we determined these “core components” before moving on to further design

These include:

- Microcontroller
- Camera sensor

Component selection

Microcontroller

Requirements:

- Has hardware for BLE, I2C, SPI, Clock generation, and GPIO
- Support big enough RAM to host at least 1 1920x1080 image (~3MB), best to support external RAM
- Has hardware to receive camera parallel data input
- Has low sleep power draw
- Cost efficient

Our pick:

- ESP32-S3 (Later changed to ESP32-D0WD)

Component selection

Camera

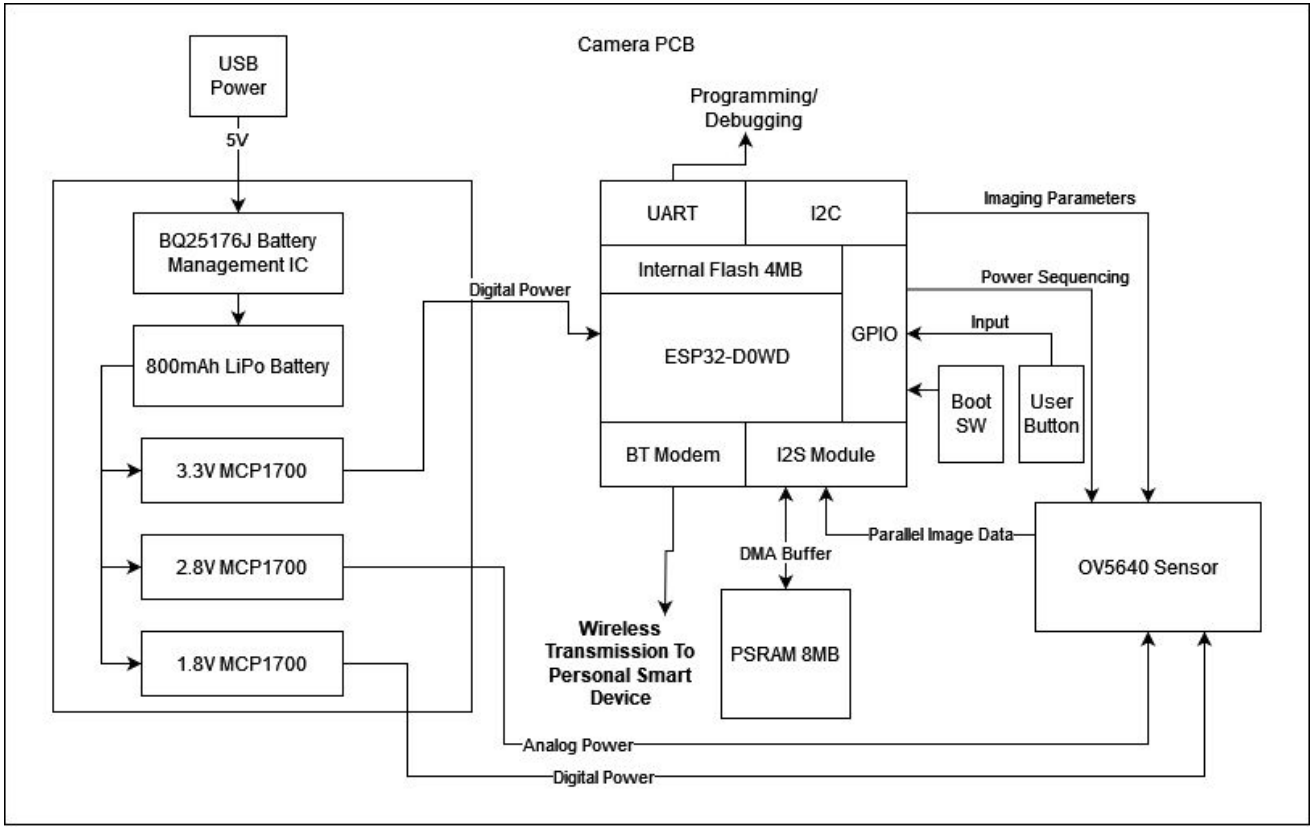
Requirements:

- High definition, support at least 1920x1080 resolution
- Works with our microcontroller
- Advanced image processing on-board
- Has low sleep power draw
- Cost effective

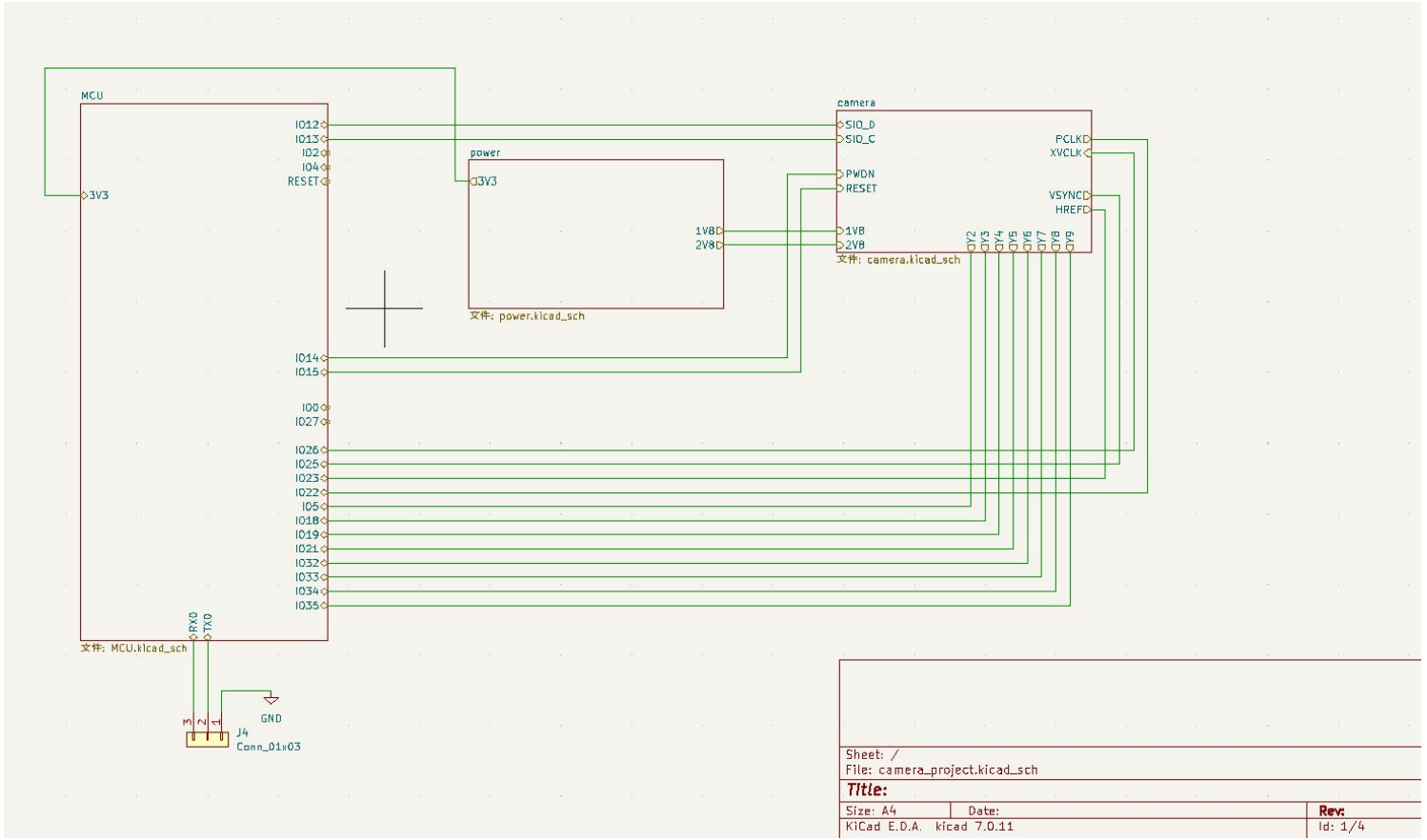
Our pick:

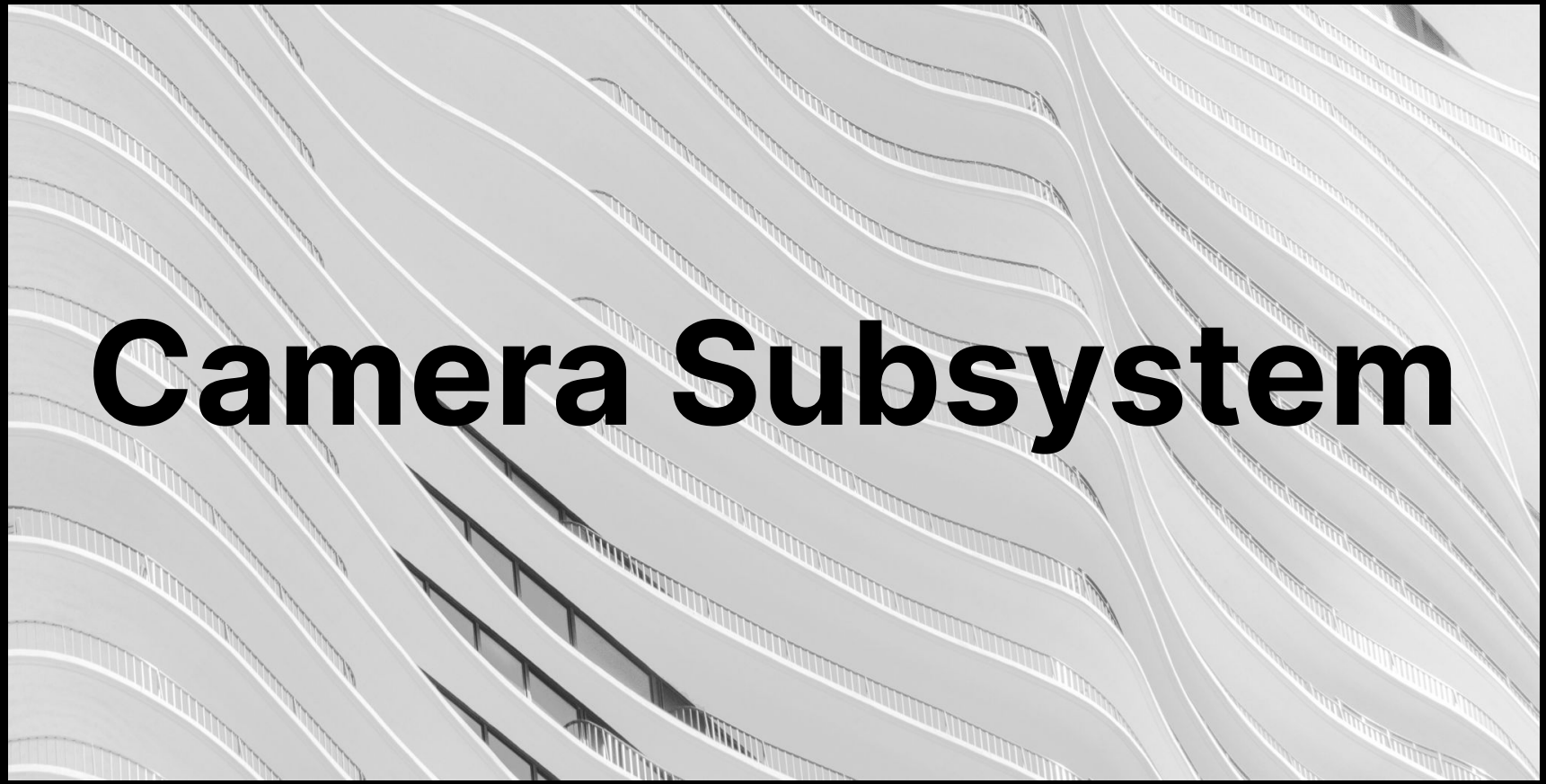
- OmniVision OV5640 with fisheye lens (~$5 ea.)

Hardware


Camera PCB
USB Power
5V
BQ25176J Battery Management IC
800mAh LiPo Battery
3.3V MCP1700
2.8V MCP1700
1.8V MCP1700
Digital Power
Programming/ Debugging
UART
I2C
Imaging Parameters
Internal Flash 4MB
Power Sequencing
ESP32-D0WD
GPIO
Input
Boot SW
User Button
BT Modem
I2S Module
Parallel Image Data
DMA Buffer
OV5640 Sensor
Wireless Transmission To Personal Smart Device
PSRAM 8MB
Analog Power
Digital Power

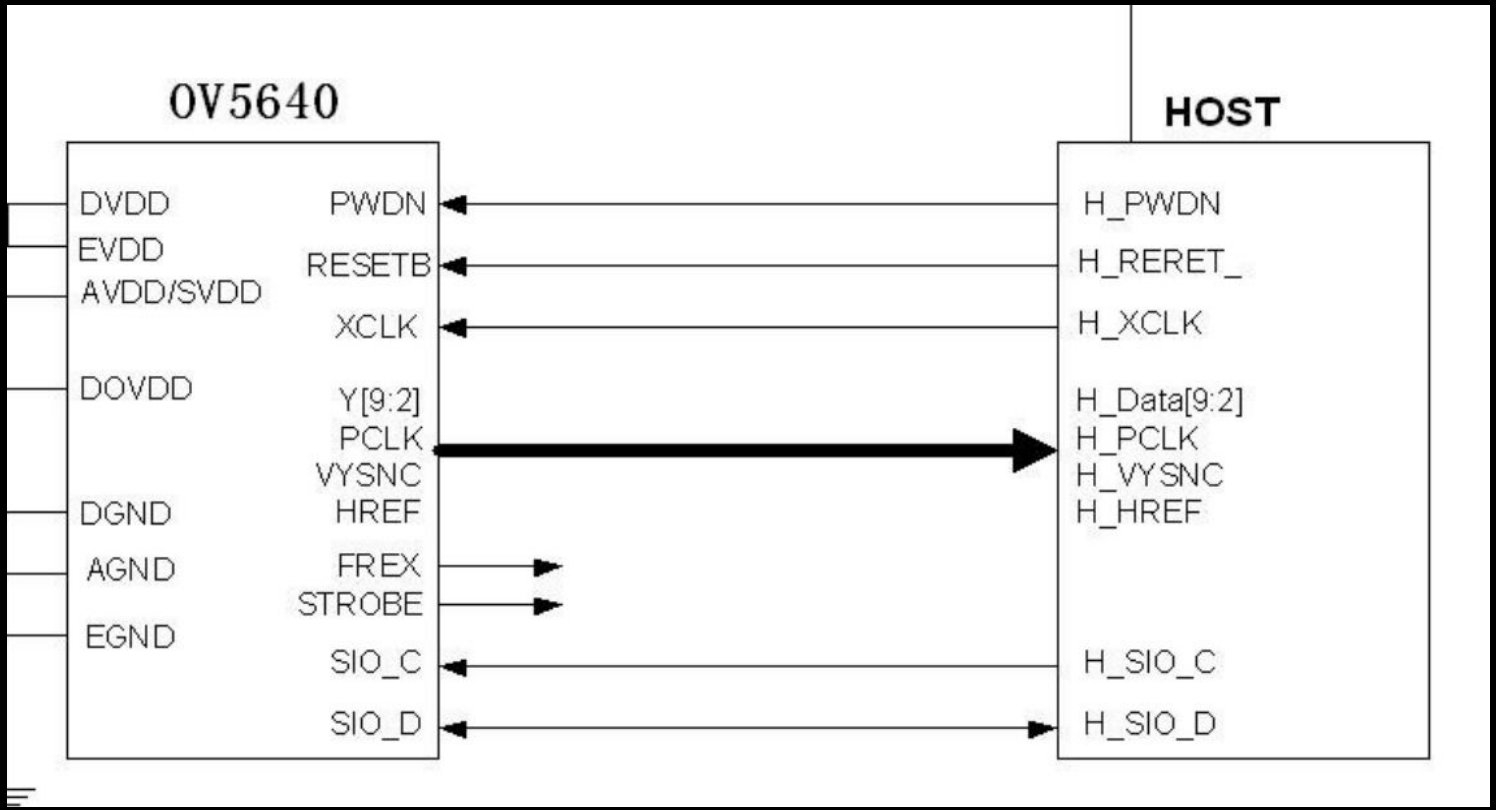
Hardware


MCU
IO12
IO13
IO2
IO4
RESET
3V3
IO14
IO15
IO0
IO27
IO26
IO25
IO23
IO22
IO5
IO18
IO19
IO21
IO32
IO33
IO34
IO35
RX0
TX0
文件: MCU.kicad_sch
power
3V3
1V8
2V8
文件: power.kicad_sch
camera
SIO_D
SIO_C
PWDN
RESET
1V8
2V8
PCLK
XVCLK
VSYNC
HREF
DY2
DY3
DY4
DY5
DY6
DY7
DY8
DY9
文件: camera.kicad_sch
GND
J4
Conn_01x03
Sheet: /
File: camera_project.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 7.0.11
Id: 1/4

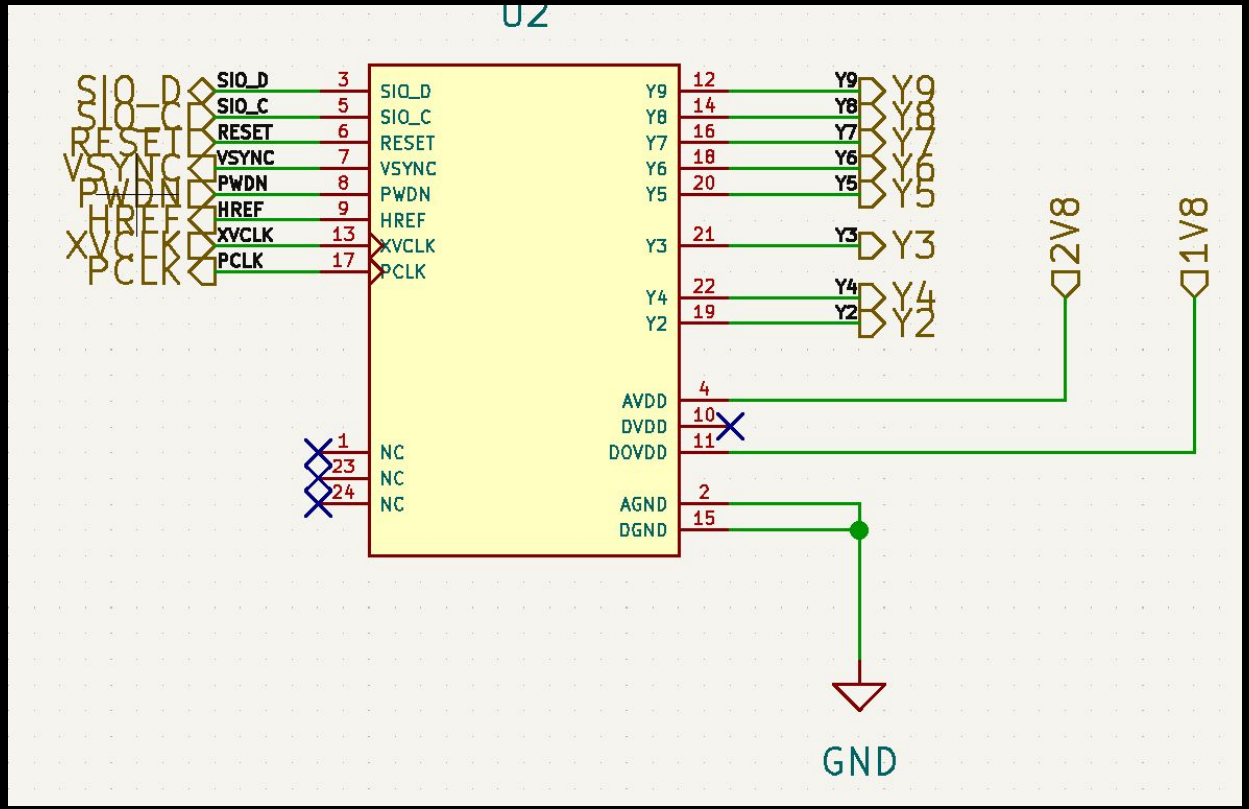


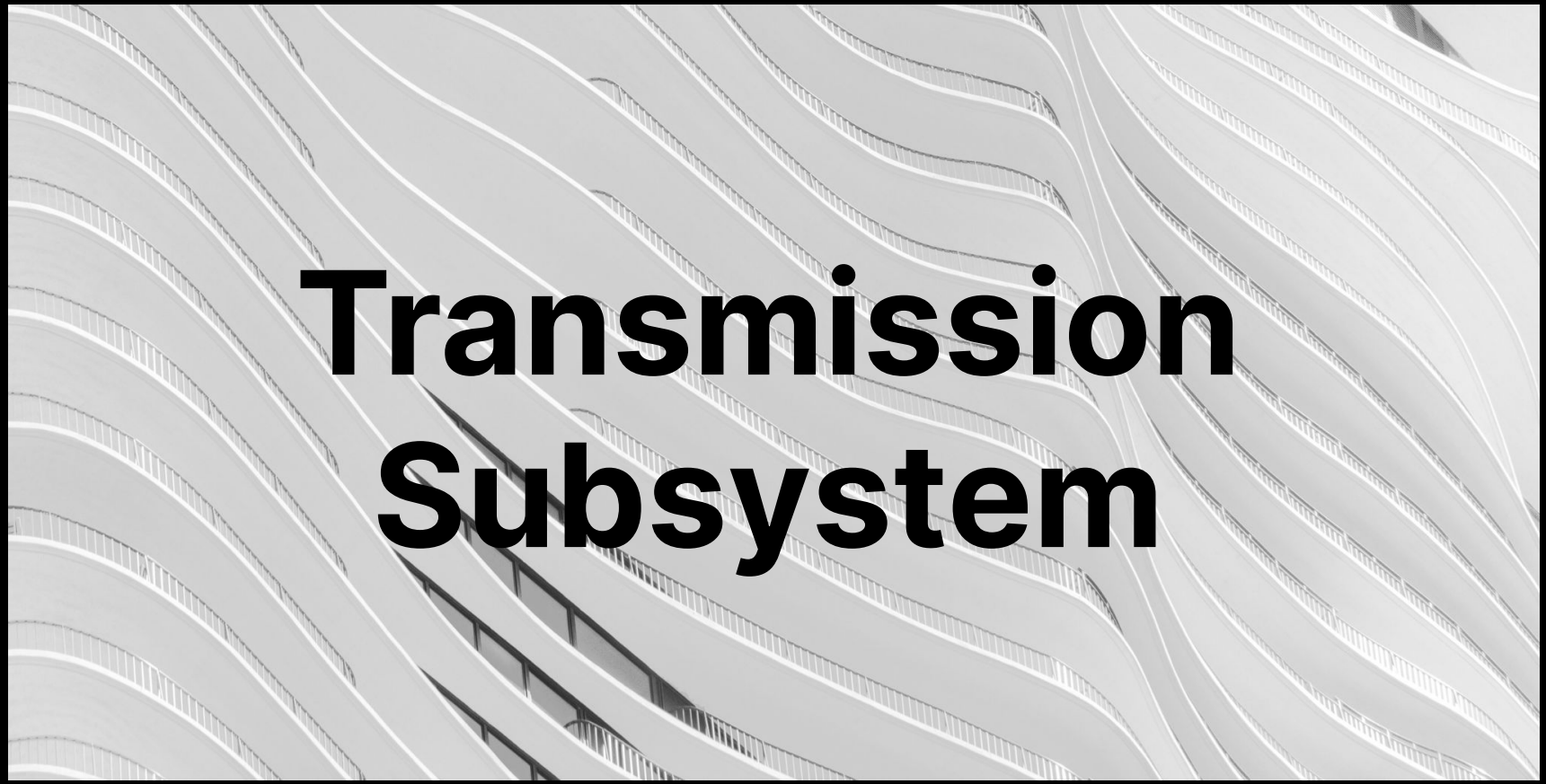
Camera Subsystem

Camera Subsystem



Camera Subsystem




Transmission
Subsystem

Transmission Subsystem

SnapLog -> host machine:

- Wirelessly transmits image from cache to host machine
- Transmit self-designed transmission protocol on top of BLE

Host machine -> SnapLog:

- Allows user to set image capture interval through our GUI
- Reports any packet loss
- Synchronizes the transmission with our protocol

Transmission Protocol pt.1

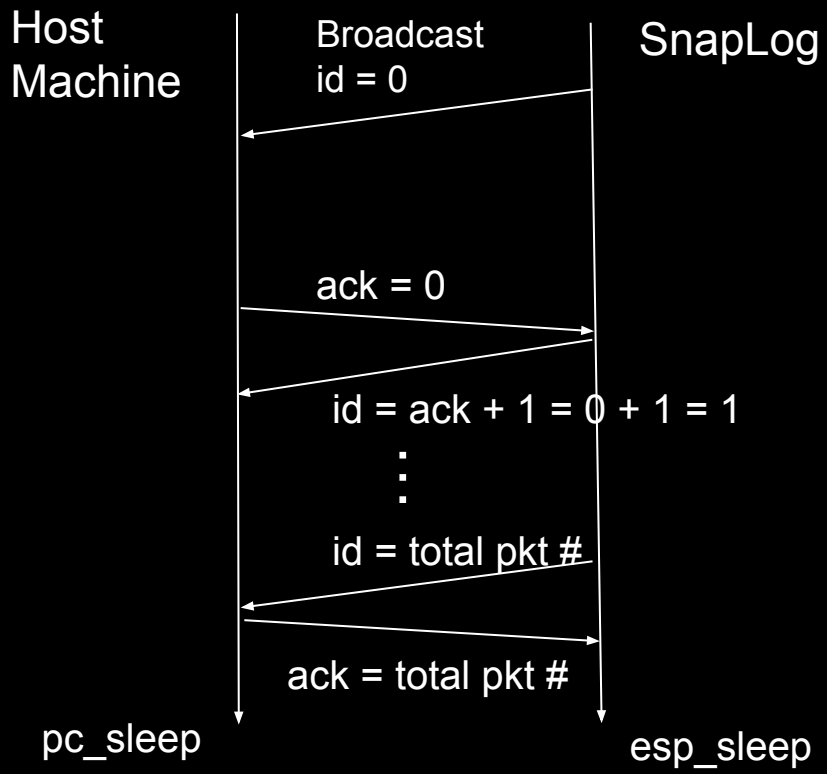
Packet Header (9B)

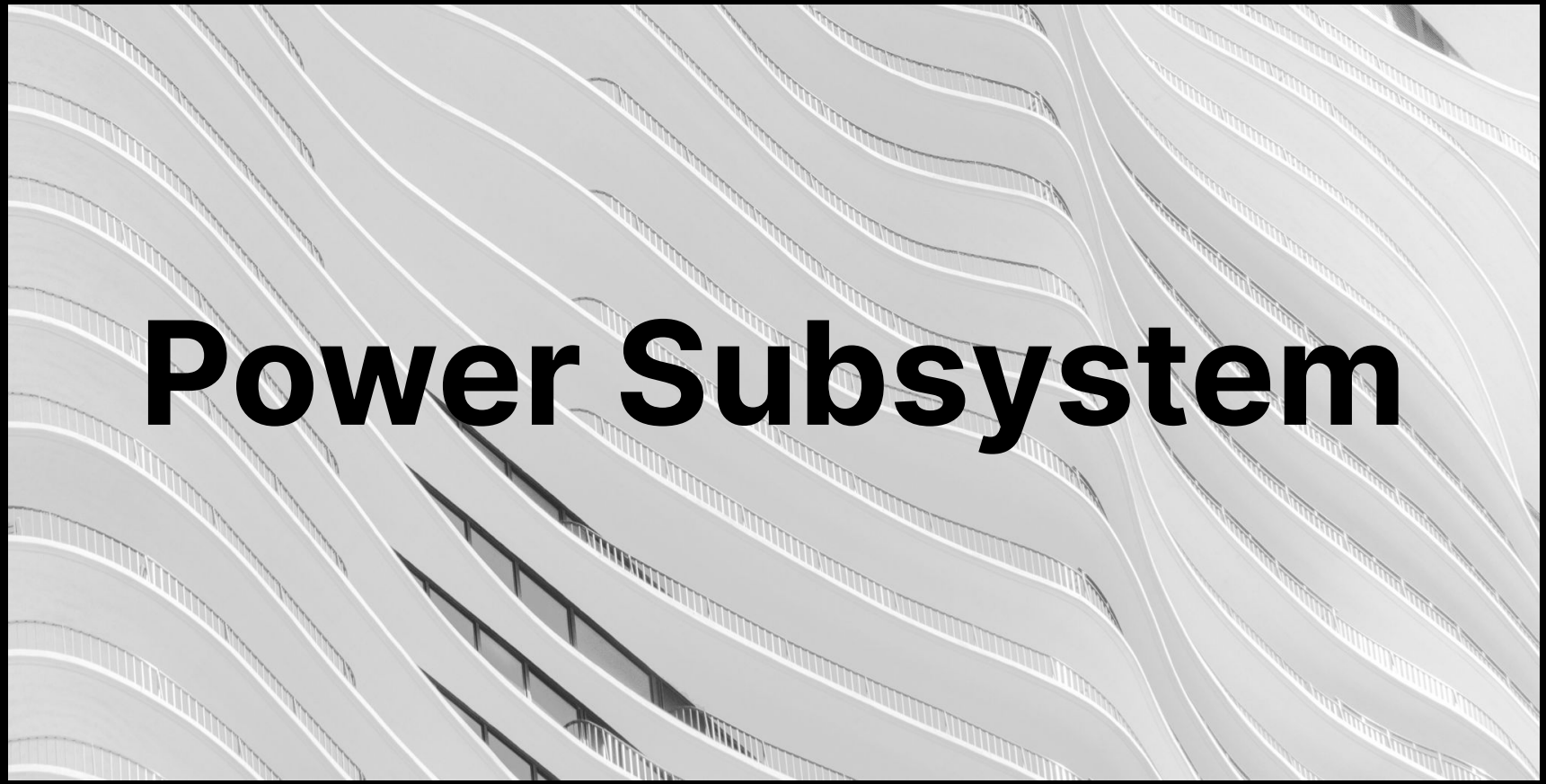
Data

Magic Starter (1B)	Packet Size (2B)	Data Size(2B)
id Number (2B)	Total Packet Num (2B)	
Data (500B)		

Total (512B)

Transmission Protocol pt.2





Power Subsystem

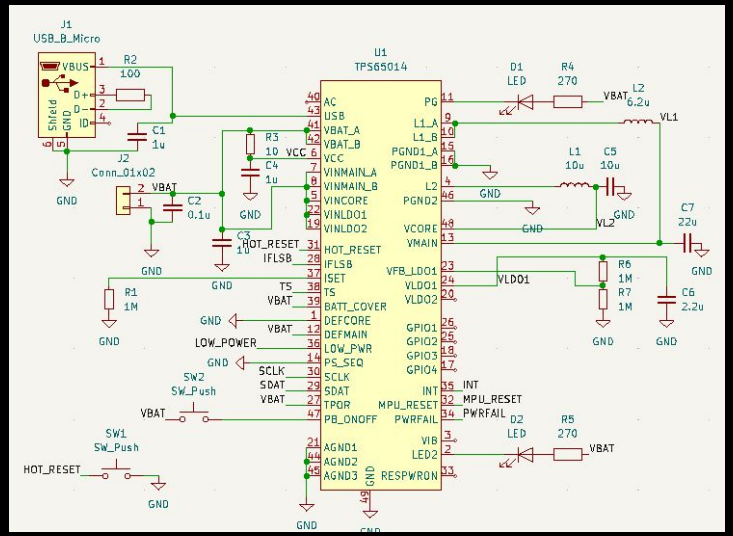
Power Subsystem

Need to:

- Send correct voltages to components
- Control power sequencing
- Battery management
- Enough to pass current for every component
- Power sequencing
- Handle sleep

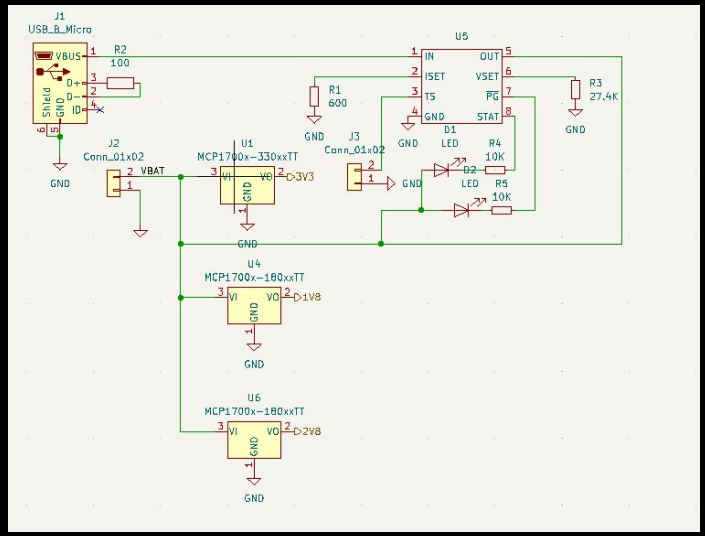
Power Subsystem

Initial design:



- All-in-one solution, large footprint
- Very expensive (~$30 per design)
- Extra features we don’t need

Revised design:



- Simple and functional
- Cheap (~$3 per design)
- Check all the boxes

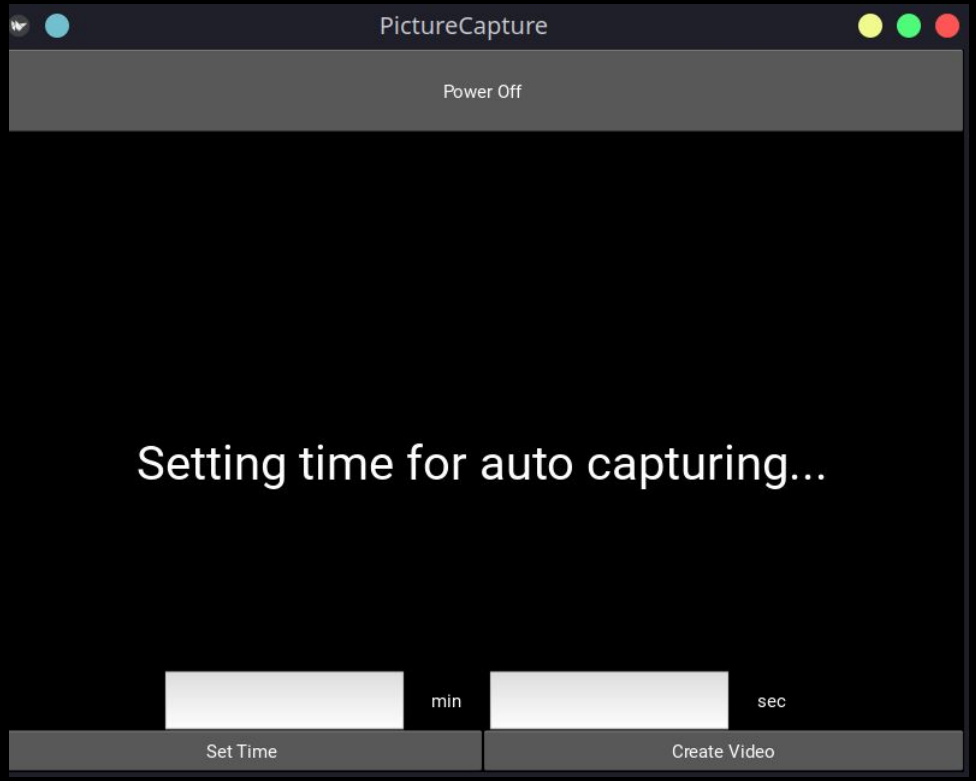
Power Subsystem

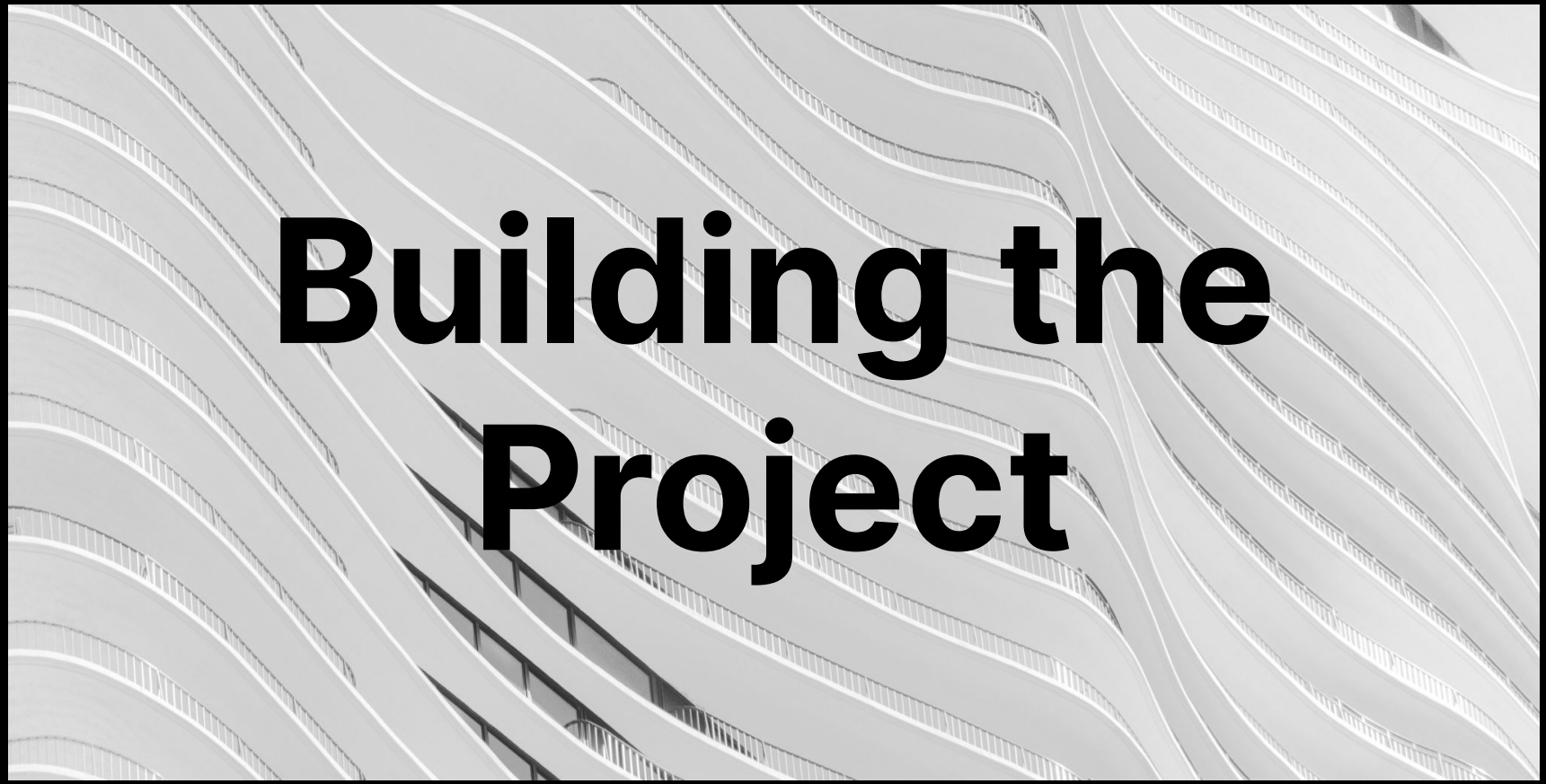
Power Sequencing

- Initial design: TPS65014 Power Management Chip handles everything
- Revised design: Software controlled by ESP32

Graphical User Interface (GUI)

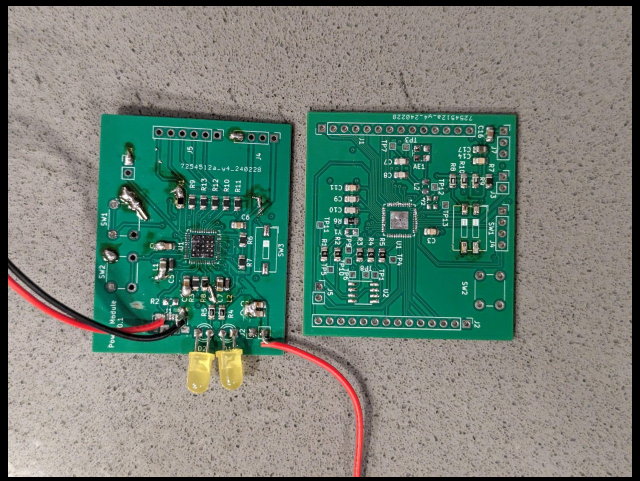
Graphical User Interface





PCB design

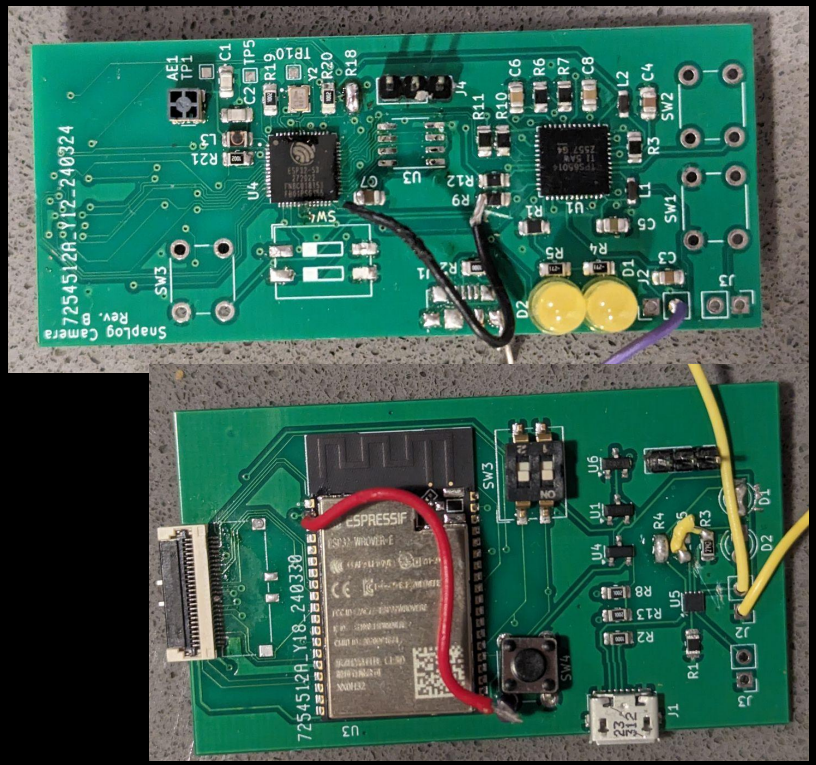
Rev A: Bring-up boards



- Individual boards for each subsystem, designed to be connected with wires
- Chip removed and reused in next revision for the sake of cost
- The purpose was to test each individual subsystem and PCB design to rule out hardware design errors
- Major issues ruled out: Wrong connections, Ground network issue in combined PCBs

PCB design

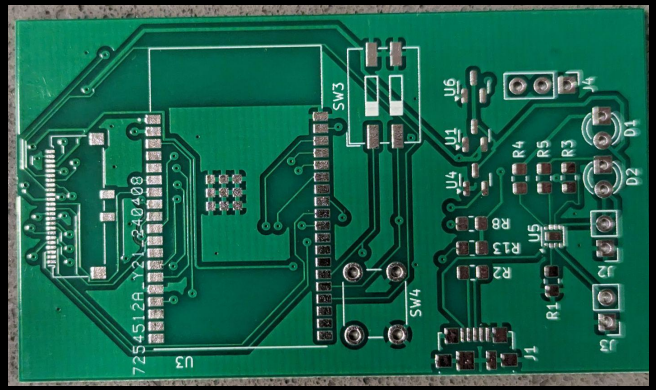
Rev B: Prototype



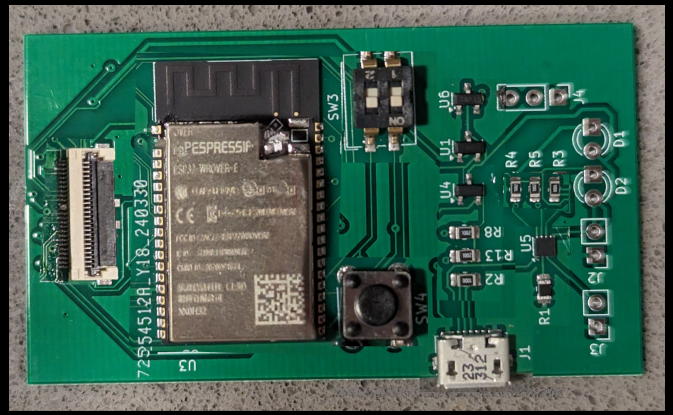
- Combined each subsystem on one PCB
- The purpose of this board is to verify the integration of subsystems
- Major issues fixed: Swapping PMIC for cost, charge controller wrong orientation, floating EN pin causing unable to boot, Camera connector wrong voltage pin orientation, microcontroller package swap

PCB design

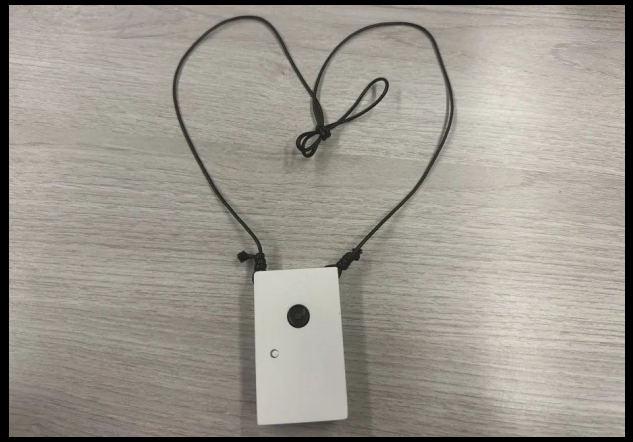
Rev C: Bug fixes, fully functional

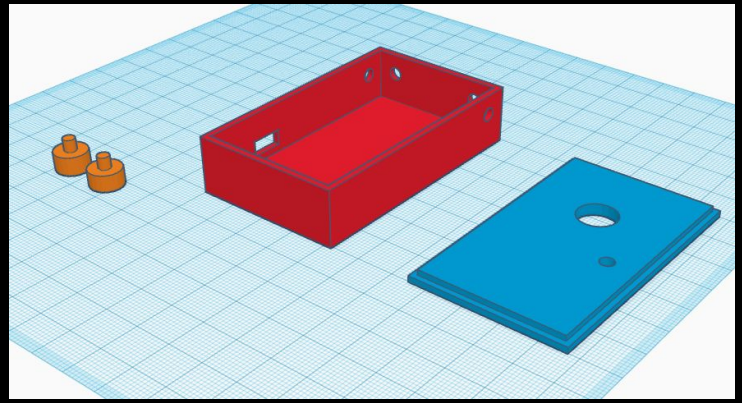


- Final board fixed all previous issues
- Assembled and working, as we are seeing in the demo.
- We forgot to take a photo for the final board before sealing the enclosure so here is a photoshopped image from a blank final PCB demonstrating what it looks like



Professional Enclosure





- We professionally designed an enclosure to finish off our product in a industry standard CAD software
- We 3D printed our enclosure and assembled with consumer electronic grade adhesives to ensure repairability


Challenges

Challenge 1: Time

- The time for this project is very limited (~1.5 month), while logistic of this class is slow (turnaround time for each revision ~1.1 week)

Our solution:

- Plan ahead: We included 1 week of “buffer time” in case the unexpected happens
- Ignore class schedule: As we already know ordering with class is slow and unreliable, we ordered everything out-of-pocket and ahead of time.

Challenge 2: Hardware debugging

- Sometimes our circuit doesn’t work and we don’t know why.

Our solution:

- Apply hardware debugging techniques to alter PCB (jumping wires, cutting/scraping traces, injecting power, etc.) to hunt down the issue and reuse the same PCB to avoid one more order in uncertainty
- Changing components to move more weight-lifting to software instead of hardware to make changes wayyy faster (i.e. power sequencing)

Challenge 3: Work life balance

The ECE classes are taking away our personal lives.

Our solution:

- Definitely, NO WORK DURING SPRING BREAK!!
- Avoid working on Saturday, Sundays if possible.
- Work smart not hard.


Conclusion

Conclusion

- Overall, our product successfully achieve our the functionality that support our users' need.

- In the future, we will more feature including auto focus…

Demo


Thank you for
tuning in!