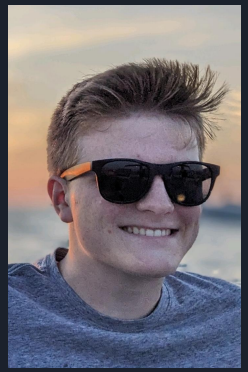
KIPS Mk. II

Group 16

Meet the Team

Scott Valentine
EE
Hardware



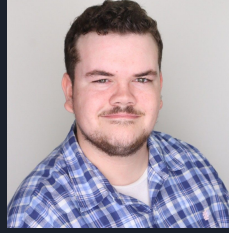
Jonathan Meyers
CpE
Software

Jessenia Argueta
CpE
Software

Harrison Mills
CpE
Software

Distribution of Work

	Sensors	PCBs	High Level Software	User Interface
Primary	Jessenia	Scott	Jonathan	Jonathan
Secondary	Harrison		Harrison & Jessenia	

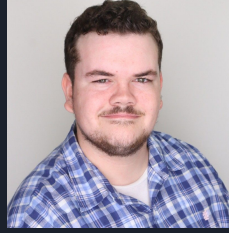


Motivation and Background

Inspired by the pocket PC and other modern smart devices, our group wanted to create a portable PC device with higher functionality, versatility, and modifiability than today's commercially available devices.

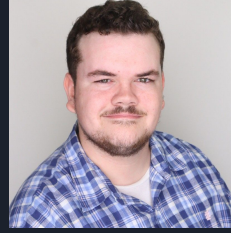
The Knights Information Processing System Mark II (KIPS Mk. II) is a wearable, wrist-mounted computing device, designed as an extensive improvement and overhaul of the earlier KIPS Mk. I prototype.

KIPS Mk. I



- Completed for 2024-2025 UCF IEEE Internal Projects Competition
- Rough draft of K.I.P.S.
- Features:
  - Touch Screen
  - I/O port
  - Basic battery power
  - Rough mobile UI (Rust & Python)

KIPS Mk. II



- Designed as a portable personal computer with mobile and desktop use.
- Battery powered for use on the go
- Peripheral integration (mouse, keyboard etc.) in desktop mode
- Array of various sensors to provide various environmental readings
- AM/FM radio capability
- LED flashlight
- D-pad control for ease of use in mobile mode
- etc.

Goals and Objectives

- Our goal with this project is to design an affordable and portable personal computer that can function as a wearable smart device as well as a dedicated PC, providing the user with immediate information about themselves and the environment.
- Due to the multifaceted nature of this project we have outlined the following goals, followed by objectives, categorized into hardware and software domains.

Hardware Main Goals

- Design a wrist wearable device that serves as a marriage between a pocket PC and a smart-watch.
- Our prototype will be able to facilitate the real-time measurement of the Heart Rate
- Our prototype will be able to measure Temperature, Humidity, and Daylight in real time
- Enable the user to use battery power and wall power to operate the device.
- Allow the user to display the contents of our device.
- Enable user navigation of KIPS Mobile Mode via directional pad (D-Pad)
- Enable touchscreen capability
- Design a lower profile shell than MkI and wrist strap while ensuring even weight distribution.
- Incorporate an LED flashlight with the use of software control and a resistor to avoid burn out.
- AM/FM Radio

Software Main Goals

- Relay accurate temperature and humidity information back to the MCU.
- Enable user vital signs to be taken and displayed through KIPS through reading of a heart rate sensor
- Allow for auto brightness setting through daylight sensor readings
- Enable mobile use through a suite of applications designed to mimic a smartwatch.
- Enable desktop use through selection of an OS and write programs to communicate with the MCU, which enables the user to make their own modifications as well.
- Integrate sensor suite with both the mobile and desktop environments.
- Enable the user to switch from mobile mode to desktop mode and vice-versa.
- Design different power modes associated with the desktop and mobile OS.

Hardware Objectives

- Develop a custom I/O board for SBC and MCU.
- Model and create a 3D printed shell to house the device.
- Battery Power
- Power Button
- Heart Rate Sensor
- Light Sensor
- Temperature and Humidity Sensor
- AM/FM Receiver
- Implement an HDMI port in the system

Software Objectives

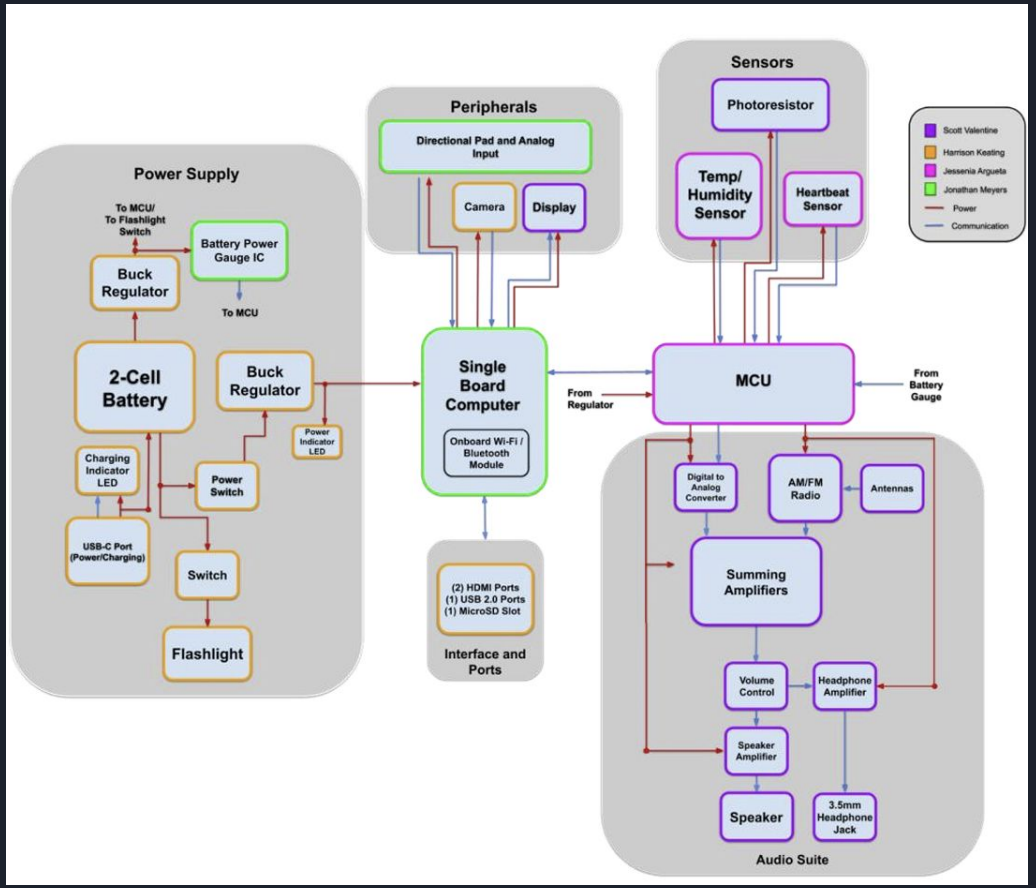
- ❖ Develop and include various mobile-oriented applications
- ❖ Write software protocol for switching between mobile and desktop operating systems

Table of Engineering Specifications

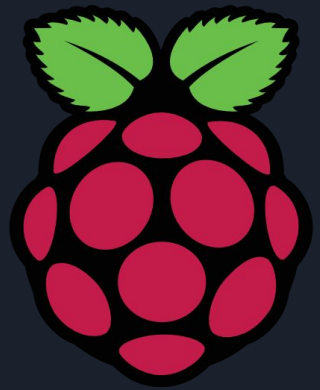
Component	Parameter	Specification
Desktop Battery Life	Time	> 2 Hours
Mobile Battery Life	Time	> 5 Hours
Dimensions	Size	~ 6 in. x 4 in. x 2 in.
Weight	Pounds	< 4 lbs.

Component	Parameter	Specification
Mobile to Desktop Conversion Time	Response time	< 20 seconds
Power Consumption	Wattage	< 10 Watts
Avg. System Operating Temp.	Temperature in fahrenheit	95°F - 120°F
Response time/ time delay	Response time	< 10 seconds

Hardware Design



PICTURE OF FINISHED DEVICE

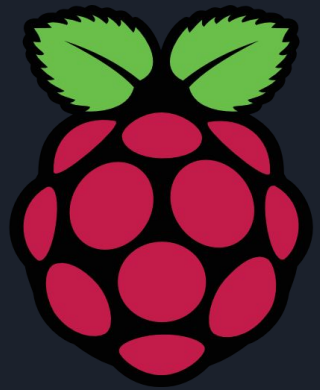


SBC, MCU, and FPGA Technology Comparison

	FPGA	MCU	SBC
OS Support	N/A	N/A	Yes
Power Draw	High	Low	Moderate/High
HDMI Support	Yes	N/A	Yes
USB Support	Yes	Limited	Yes
Boot Time	Near-instant	Fast	Slow
Price Point	High	Low	Moderate
Form Factor	Large	Very Small	Small

Hardware Selection

- SBC
- MCU
- Regulators



SBC Product Selection

	Raspberry Pi	Radxa (Rock Pi)	BeagleBone
Processor	Quad-core	Quad or Eight-core	Single, Dual or Quad-core
RAM	0 (lite), 512MB, 1 GB, 2, 4, 8, 16 GB	256MB, 512MB 1 GB, 2GB, 4GB, 8GB, 16GB, 32GB	512MB, 4GB
Onboard (Flash) Storage	0 (lite), 16, 32, 64 GB	N/A	0, 4 GB, 16 GB
USB	USB 2.0 & 3.0	USB 2.0 & 3.0	USB 2.0
Communication Protocol	UART, SPI, I²C	UART, SPI, I²C	UART, SPI, I²C
Input Current	0.7-1A, 3A	1A, 3A	1A, 3A
OS Support	Raspberry Pi OS, Ubuntu, LibreELEC*, RetroPie*, DietPi*	Radxa OS, Ubuntu	Debian, Linux, and Android Support
Price/Price Range	$45-$135	$45-$200	$55-$228

SBC Part Selection

	Pi Compute Module 4 (CM4)	Pi 5 Compute Module (CM5)
RAM (GB)	1, 2, 4, 8	2, 4, 8, 16GB
Flash Memory (GB)	0 (lite), 8, 16, 32	0 (lite), 16, 32, 64
Processor	quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	quad core Cortex-A76 (ARMv8) 64-bit SoC @ 2.4GHz
Form Factor (mm)	55 x 40 x 4.7	55 x 40 x 4.7
USB	1 USB 2.0	1 USB 2.0 2 USB 3.0
HDMI	2 HDMI 2.0	2 HDMI 2.0
GPIO	≤ 28 pins	≤ 30 pins
Cost	$30-$85	$45-$135

MCU Product and Part Selection

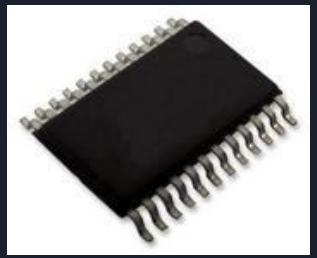
	ESP32-S3	RP2040	nRF5340	STM32U585AI
Core Type	Dual-core Xtensa LX7	Dual-core ARM Cortex-M0+	Dual-core ARM Cortex-M33	Single ARM Cortex-M33
Clock Speed (MHz)	240	133	64-128	160 (max)
Power Consumption (min-max)	~2.5µA - 260mA	~50mA - 150mA	~0.9µA - 5.1mA	~0.16µA - 3.1mA
Size (mm)	18 x 25	51 x 21	4.4 x 4	7 x 7
USB Support	Native USB-OTG	Native USB	USB	USB FS + OTG FS

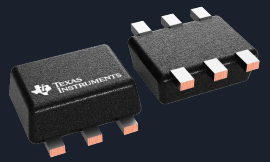
Power distribution

Component	Supply Voltage	Min Current	Max Current	Approximate wattage
SBC	5V	0.7A	3A	3.5-15W
MCU	3.0-3.6V	~2.5µA	260mA	7.5µ-0.936W
Heartbeat Sensor	3.3-5V	<10mA	10mA	33mW
Temperature and Humidity Sensor	3.1-5.5	10µA	950µA	31µ-5.225mW
Camera (SBC INCL.)	5V	100 mA	150mA	750mW
Screen (SBC INCL.)	3.3V	100mA	500mA	0.5W- 2.5W
Digital to Analog Converter	3.0V < x < 3.5V	0.5mA	13mA	1.5mW-45.5 mW
AM/FM Radio IC	2.0V < x < 5.5V	0.1mA	21.5mA	~30µW < x < 110mW
Headphone Amplifier	2.7V < x < 3.6V	0mA	10.5mA	0-37.8mW
Speaker Amplifier	2.2V < x < 5.5V	273mA	600mA	0.6006 W-3.3W
Speaker	2.83V-3.46V	0.354A	0.433A	1-1.5w
Total Wattage				Max 29.4043W

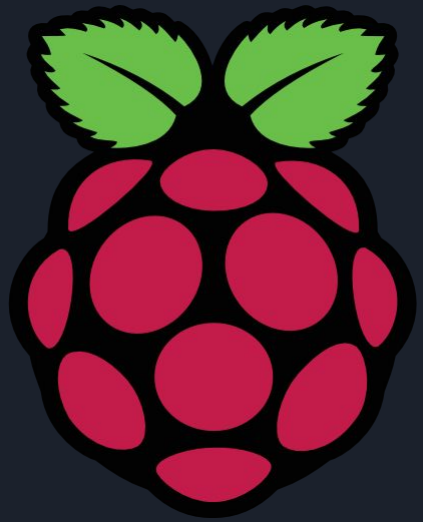
Regulator Product and Part Selection

Name	TPS566242	LTC3615
Brand	Texas Instruments	Analog Devices
Input Voltage Range	3V-16V	2.25V-5V
Reference voltage	Vin	Vin
Output voltage	0.6V-7V	0.6V-~5V
Max Current	6A	3A
Price	0.77	4.72
Efficiency	95%	94%
Switching Frequency	600 kHz fixed	200kHz to 4MHz (External resistor)





Software Selection



- Operating system
- Primary programming language (Python)
- Primary communication protocols (UART and I2C)

Operating System Selection

	Raspberry Pi OS	Ubuntu Core	DietPi
Base	Debian-Based	Debian-Based	Debian-Based
Resource Usage	Moderate, lightweight	Low-medium, higher than other options	Extremely Low, minimal overhead
Architecture	ARM (32-bit & 64-bit)	ARM SBCs / Embedded	ARM SBCs
Software Management	APT, Debian repositories	Snap packages	Menu-drive auto-installer
Update Cycle	Community / official updates available	Regular 6-month updates plus security patches	Manual/ automated installer
Ease of Customization	High, open-source	Moderate, snap packages easier to update but less flexible	Very high, minimalist

Programming Language Selection

	Python	C++	Rust	Arduino
Compatibility with ESP32	Limited (MicroPython)	Yes	Yes	Possible to
Execution Speed	Slow	Fast (compiled)	Fast	Fast
Ecosystem/ libraries	Very large (sensors, GPIO, etc)	Large (hardware, graphics, real-time, etc)	Fairly small	Limited to mainly Arduino boards
Ease of Learning	Very easy	Moderate/Hard	Hardest by far	Easy

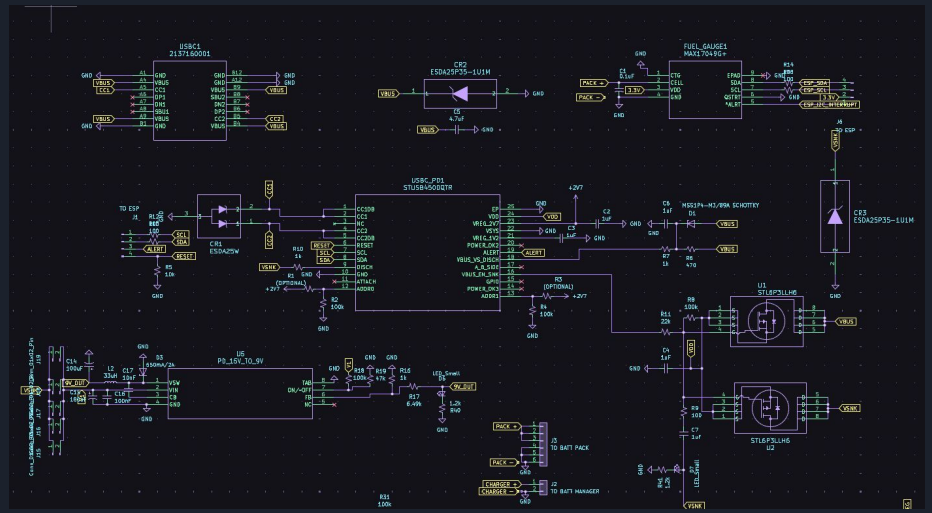
Communication Protocol Selection

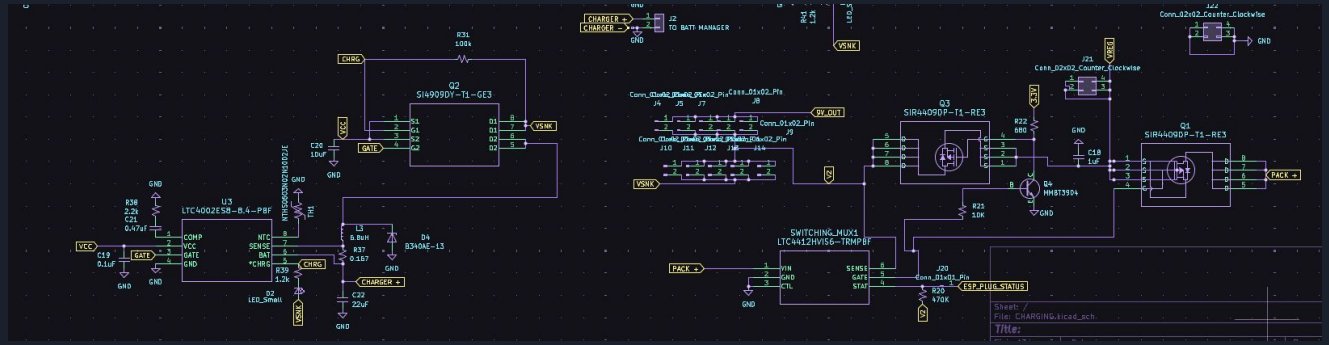
I^2C follows a synchronous, multi-master, multi-slave communication model, meaning that more than one master (controller) can exist on the same bus, and each slave device (peripheral) has a unique address.

I^2C works by having a host processor in our case a (Raspberry Pi CM4) that can tell slave devices to transmit data and also to respond to inputs

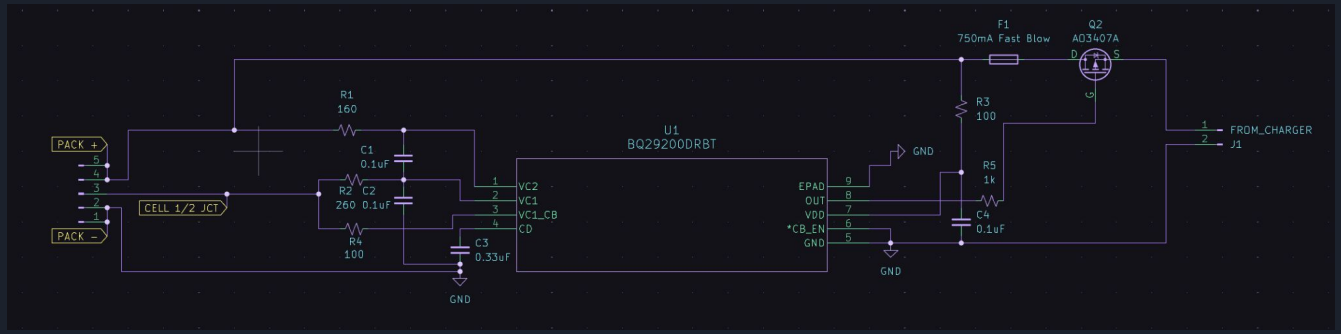
I^2C is the best choice for this project as all of our peripheral devices use it . And also for its ease of use as it requires only 1 data lane.

Power Supply Schematics - PSU

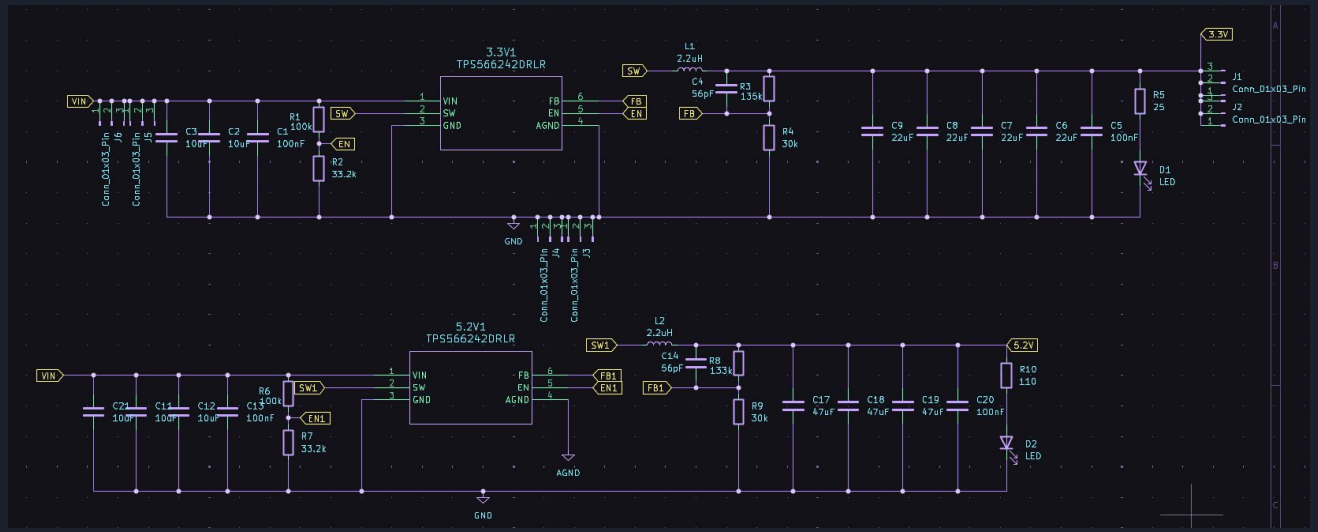




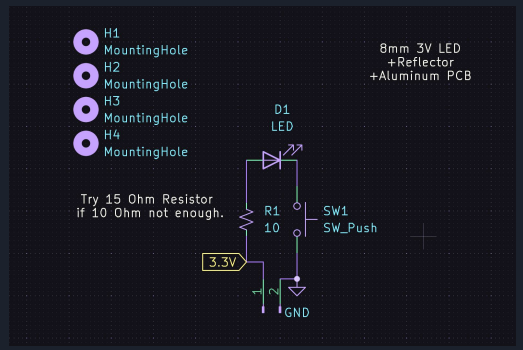
Power Supply Schematics - Battery Management System

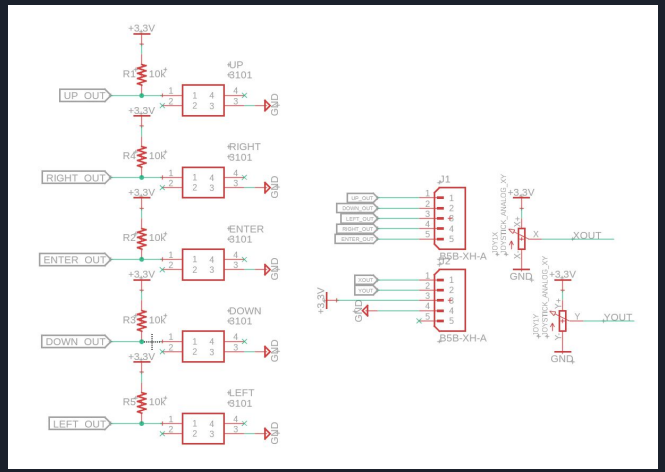


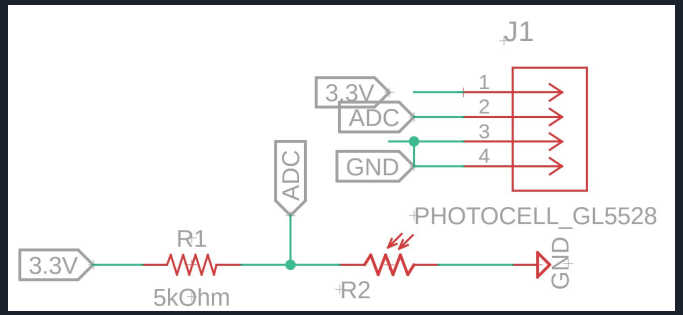
Power Supply Schematics - Regulator

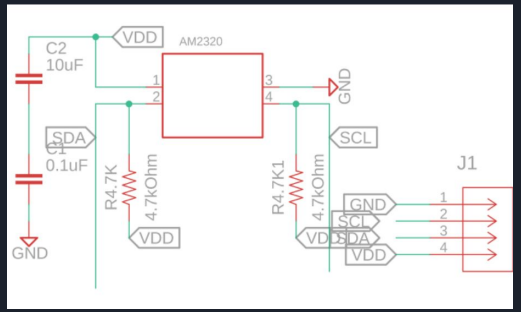


Sensors/Controls/Flashlight Schematics

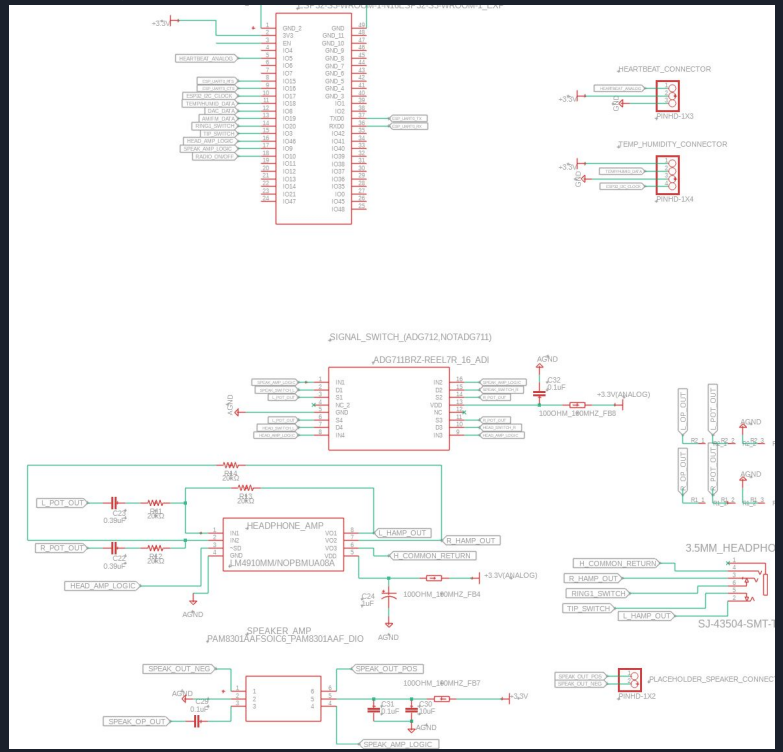


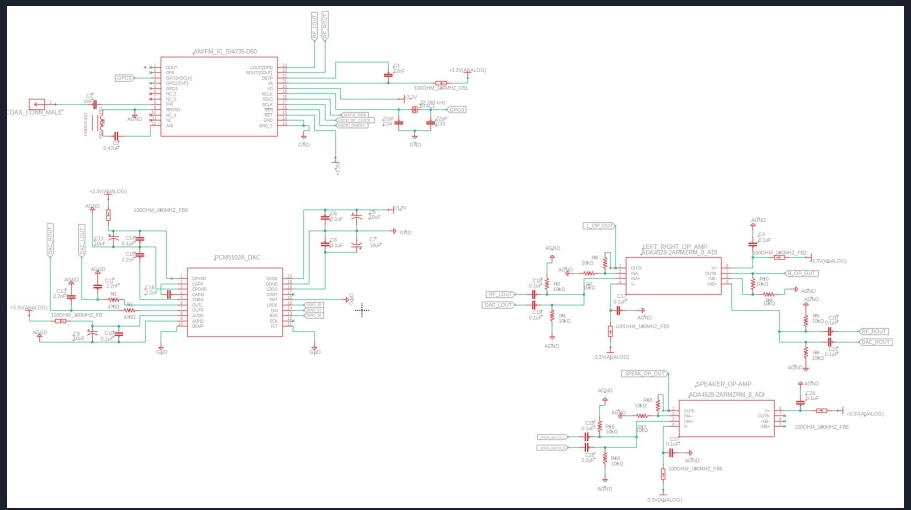




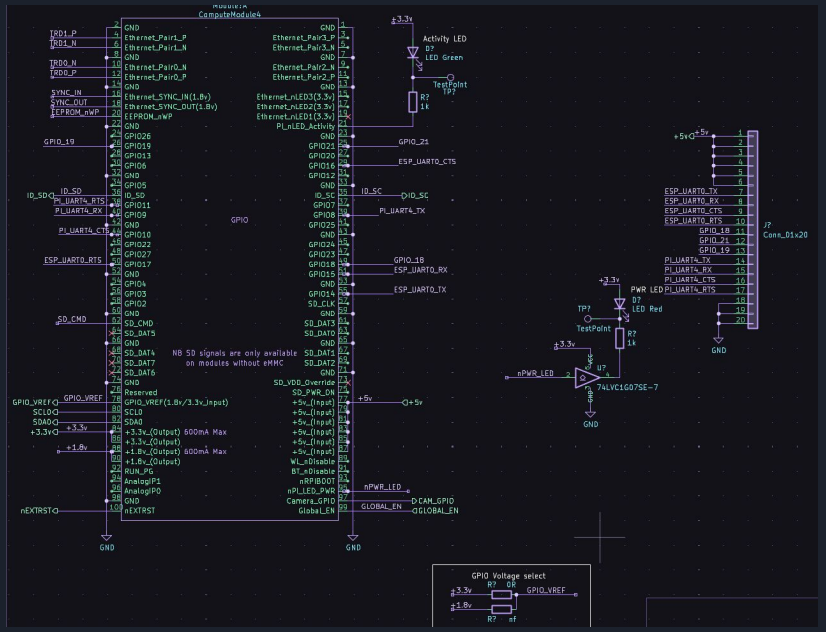


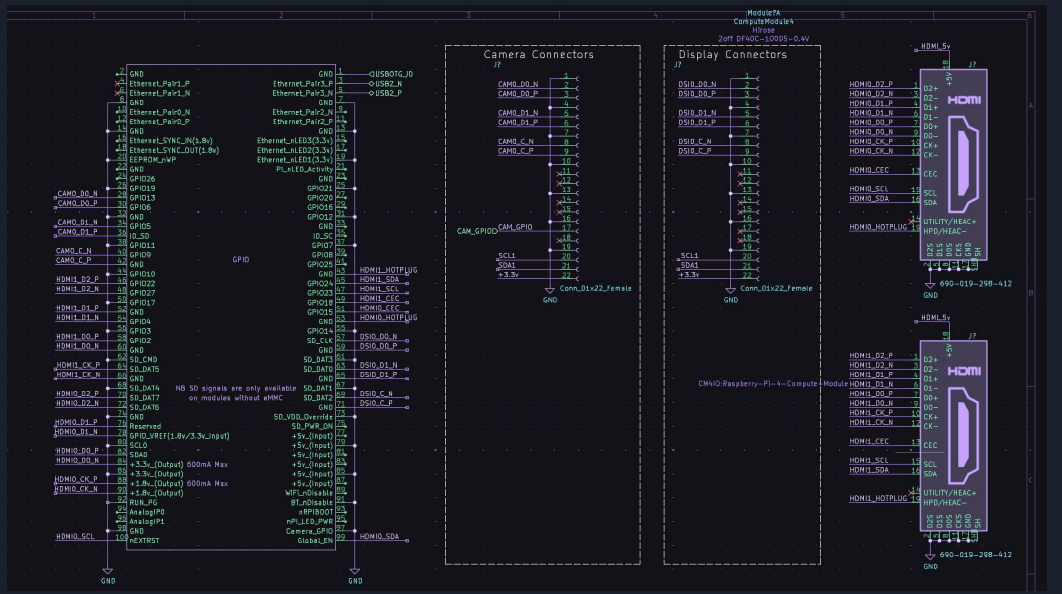
ESP/Audio Board Schematic



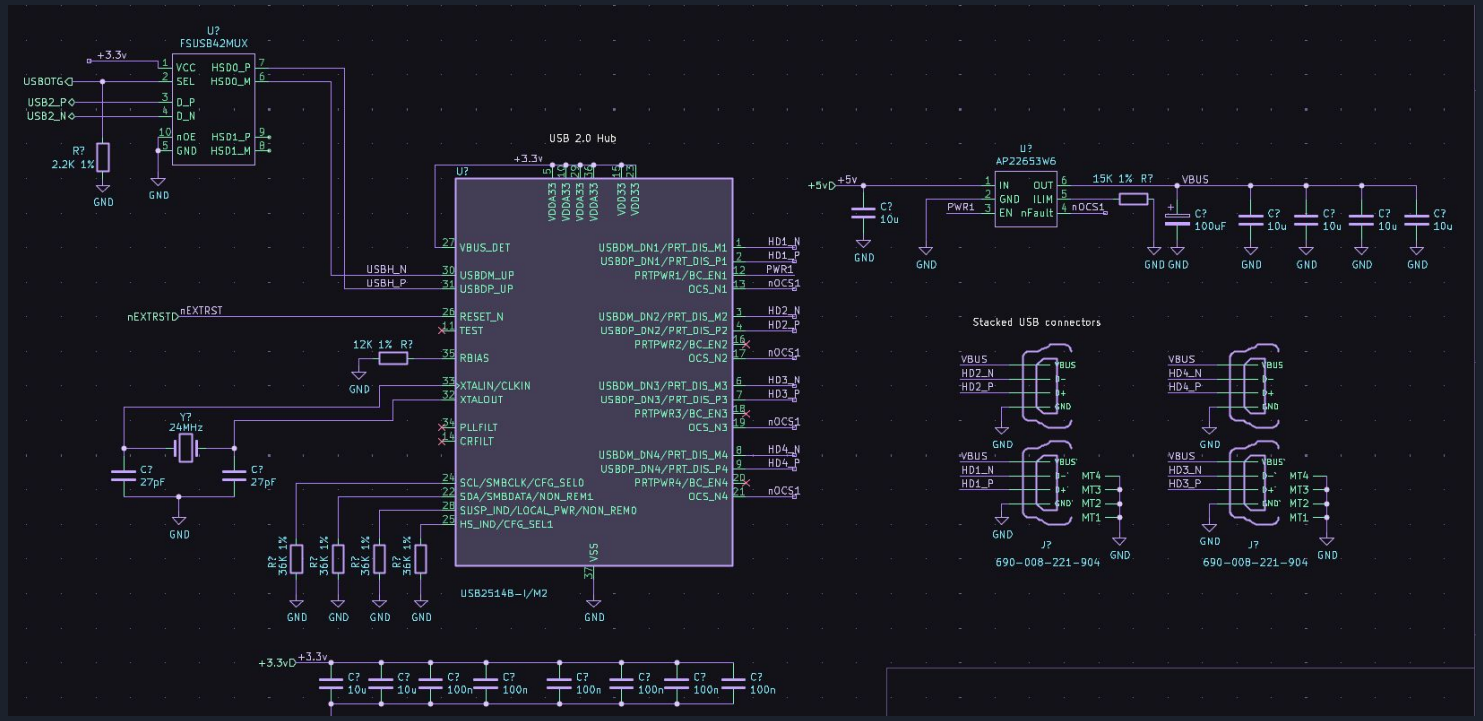


High Speed Board Schematics





High Speed Board Schematics Cont’d



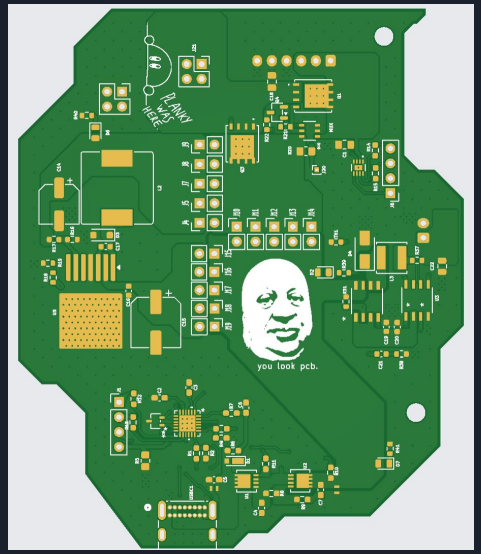
Significant PCB Design

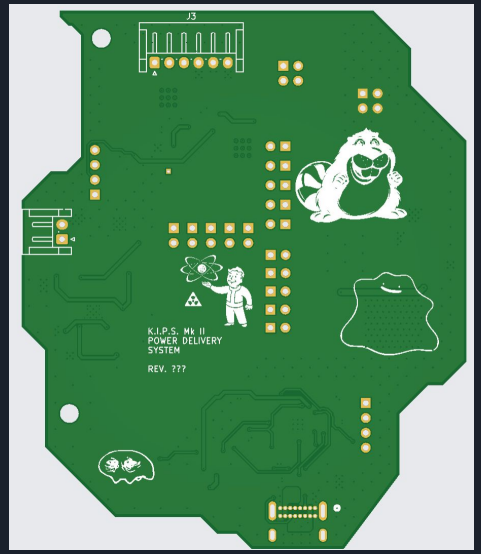
- Power Supply
  - Main PSU
    - USB-C PD Controller
    - 2S Li-ion Battery Charger
    - 15V to 9V Regulator
    - Power Switching MUX
    - Li-ion Fuel Gauge
  - Battery Management System
  - 9V to 3.3V/5.2V Regulator

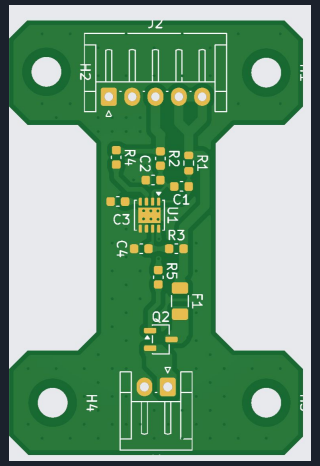
- Peripherals/Controls
  - D-Pad/Joystick
  - Sensor Boards (Temp/Humidity, Photoresistor)
  - Flashlight

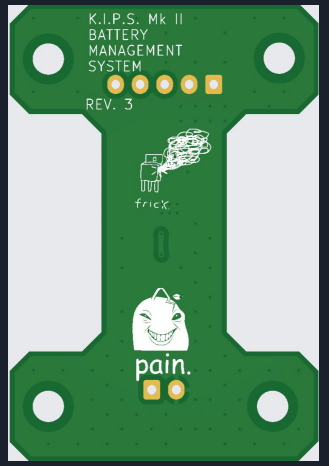
- ESP/Audio Board (WIP!)
  - ESP-32-S3 Microcontroller
  - Digital-to-Analog Converter
  - AM/FM Radio IC
  - Speaker, Headphone Jack, and Volume Potentiometer
- High Speed Board
  - Raspberry Pi CM4 Carrier
  - 2x HDMI Ports
  - 4x USB 2.0 Ports
  - 1x DSI FPC
  - 1x CSI FPC

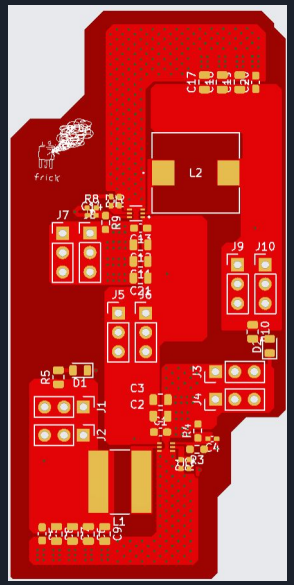
Power Supply

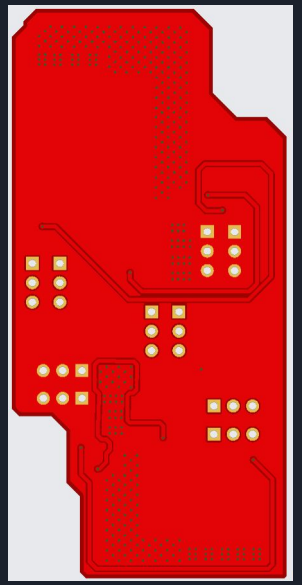


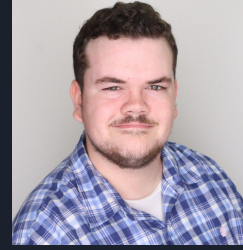




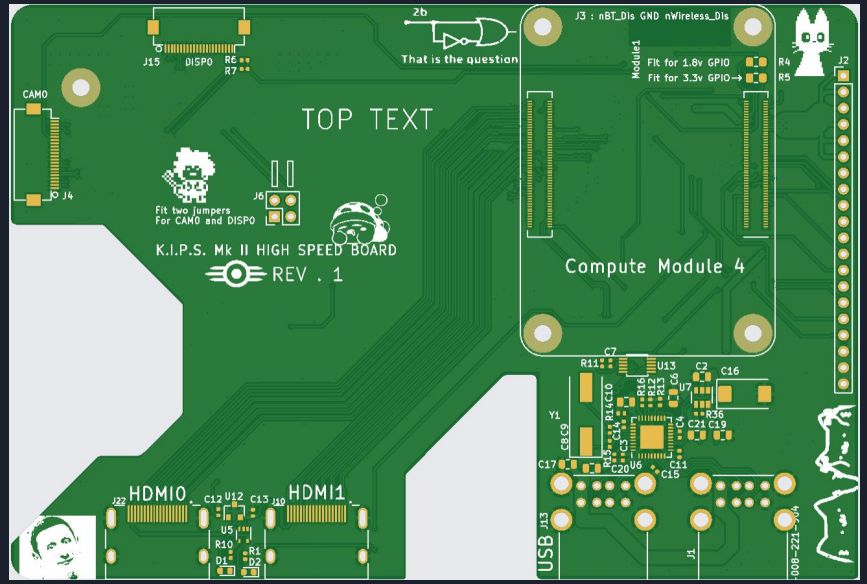


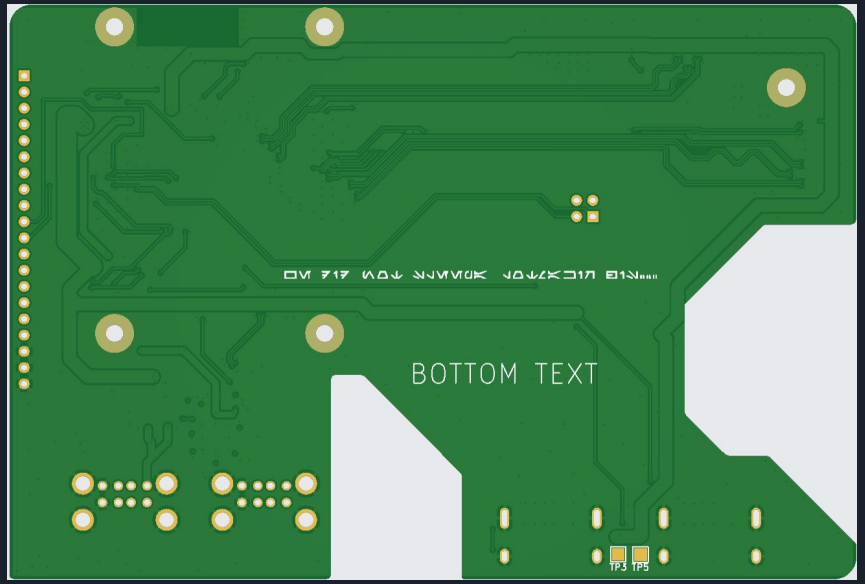




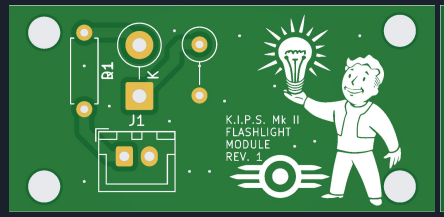


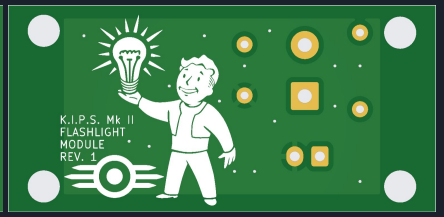
High Speed Board

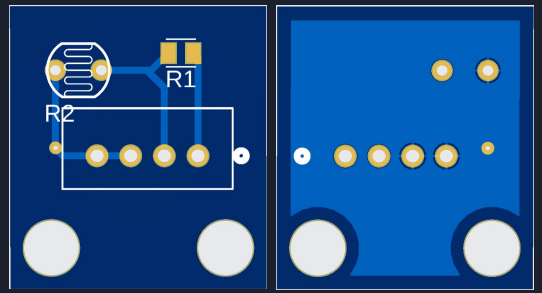


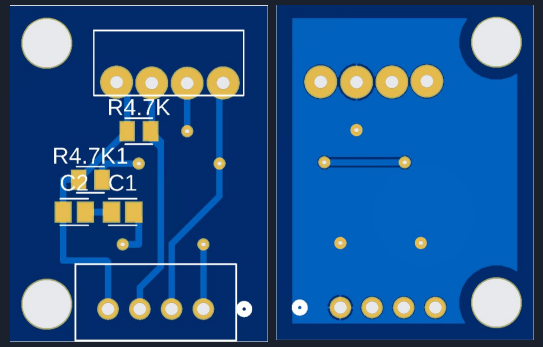


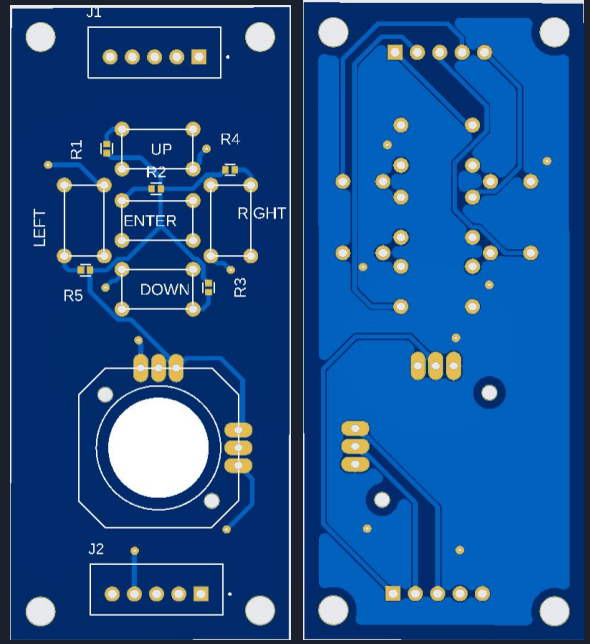
Peripherals/Controls



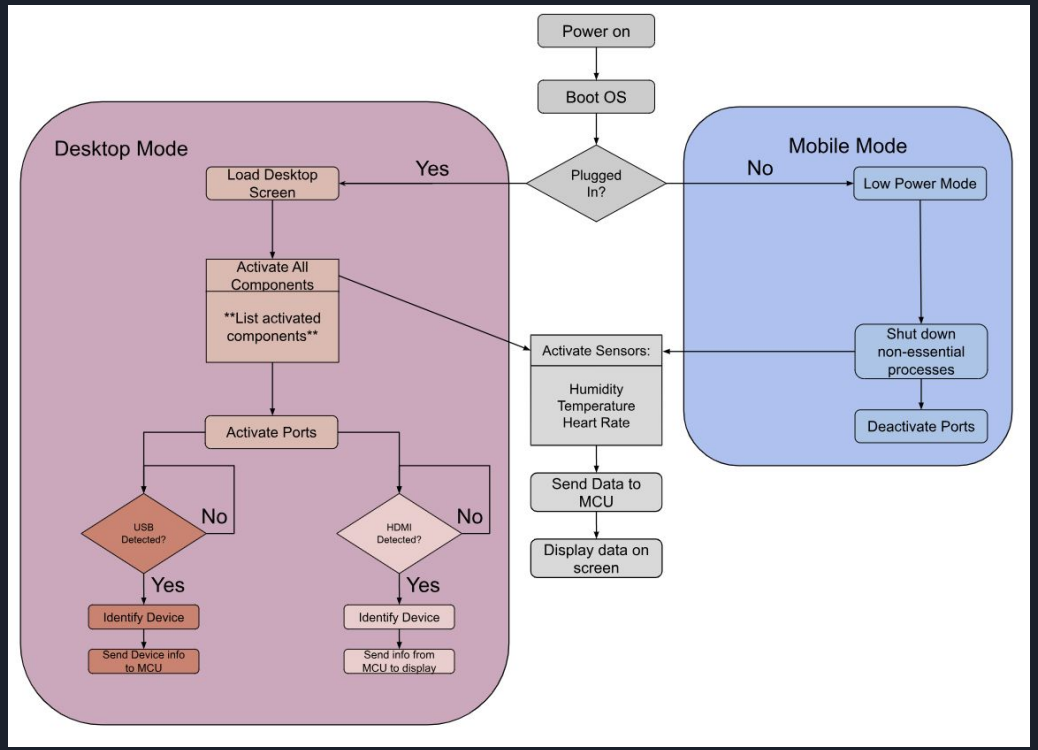


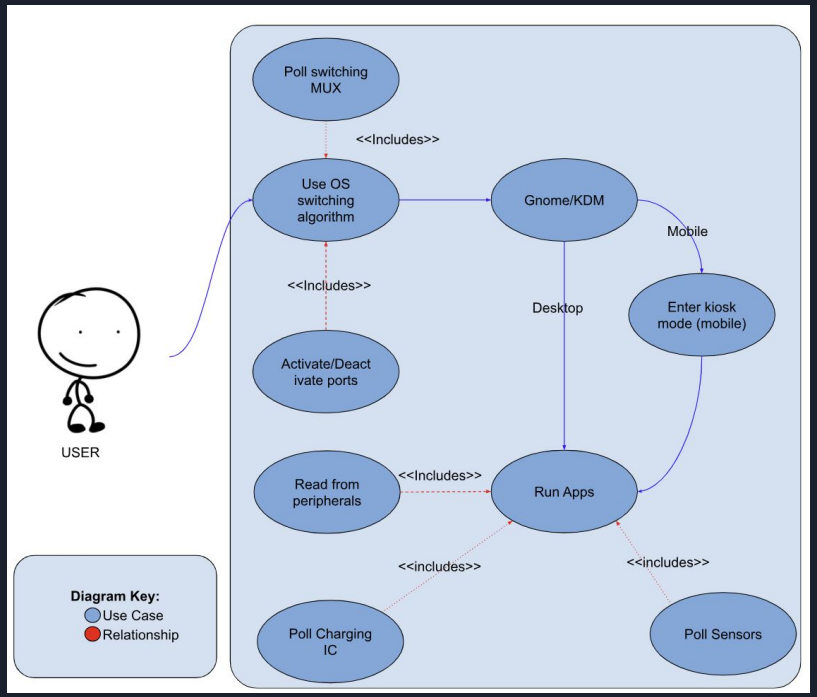




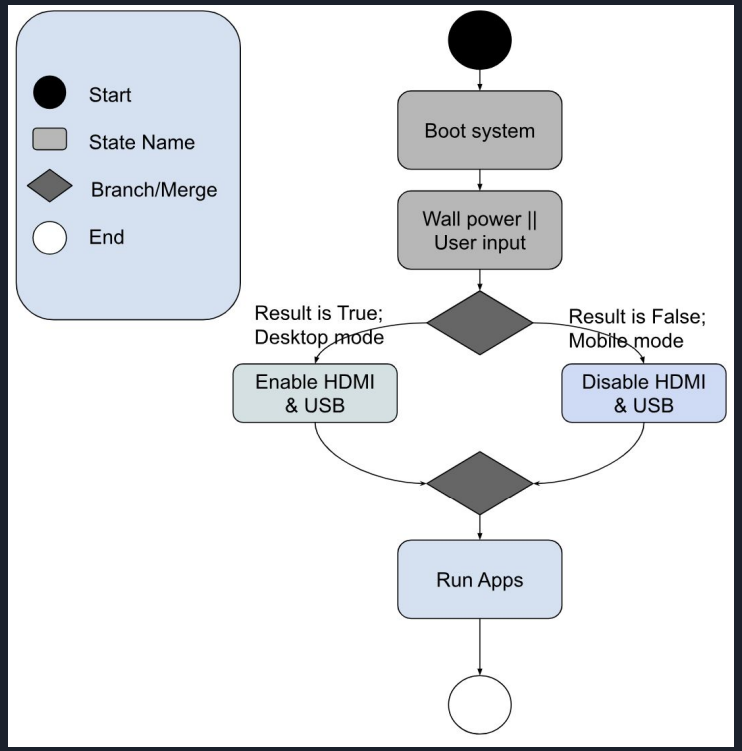


Software Flowchart and Use Case Diagram

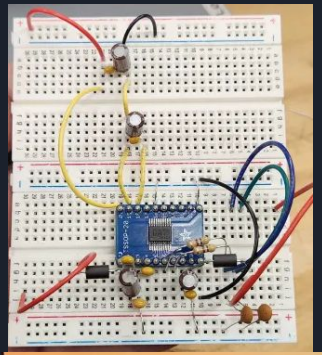


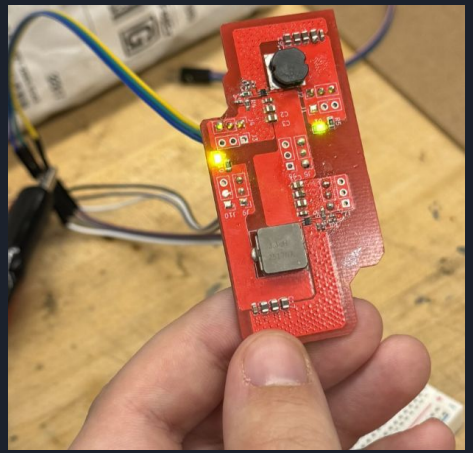


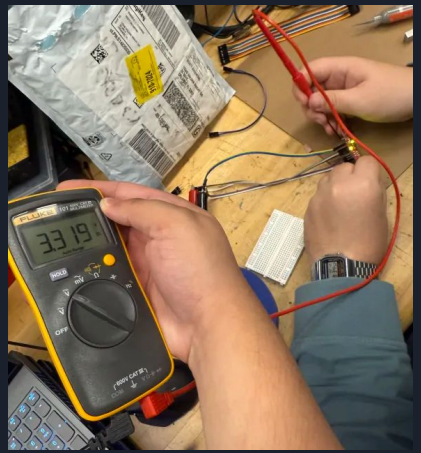
Software Activity Diagram



Hardware Testing



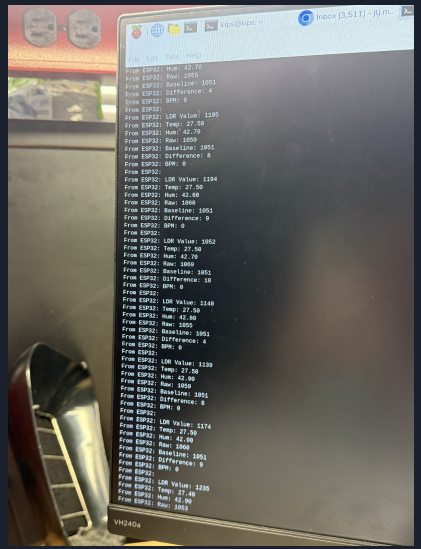


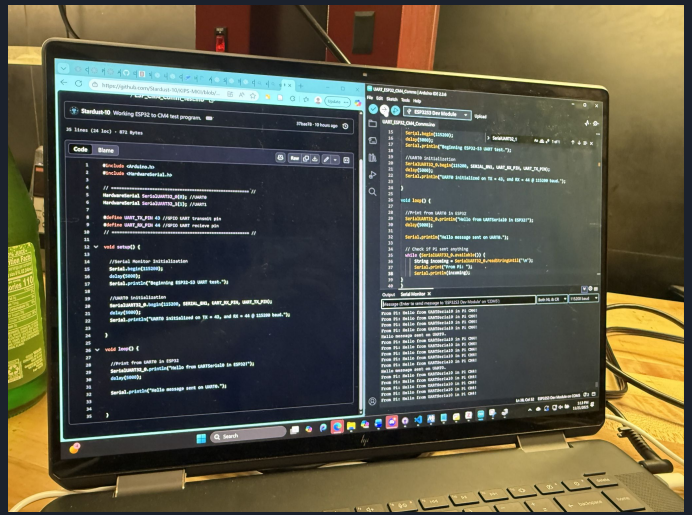


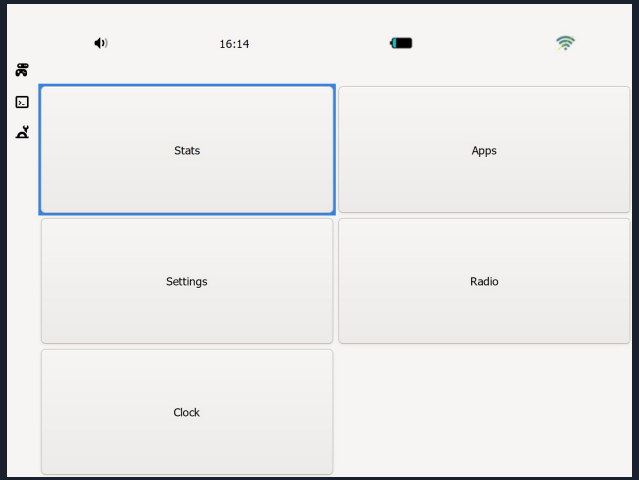
Software Testing

For the software the two major aspects that have been tested are the raspberry pi and peripherals interfaces.

Additionally, we tested the status bar icons and local clock functionality with all readings other than battery.



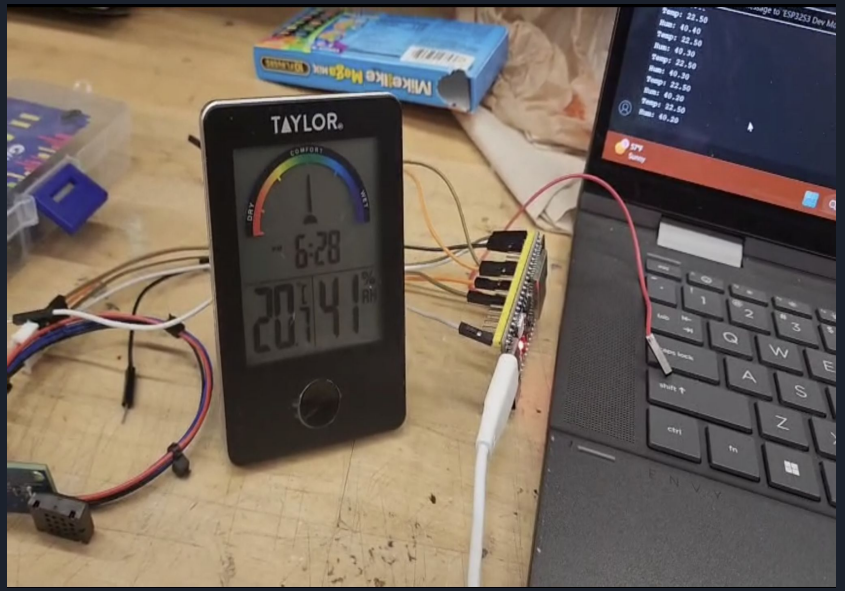




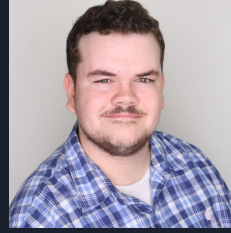
Software/Hardware Performance Evaluation

In order to test the validity of our sensor readings we measured our sensor readings against an outside source shown by this clock/thermometer/hygrometer.

Sensor readings from the sensor suite are visible in the terminal in the top-right of the included image. The corresponding readings are shown on the device in the image.



Hardware Challenges Faced



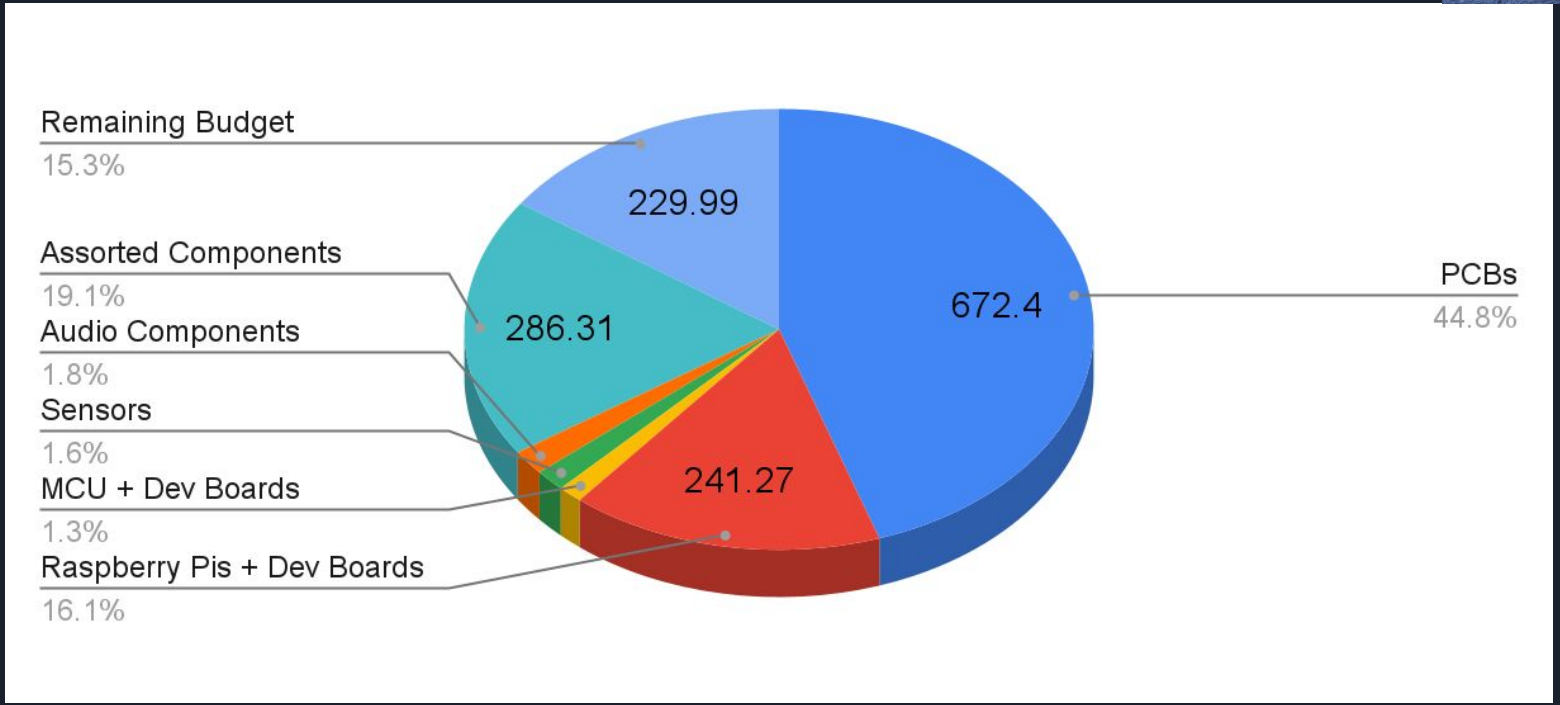
Issue	Solution
3.3V Regulator Malfunction	Trace Repair
Regulator Voltage Drop	Designed for 5.2V Instead of 5.0V
HDMI Trace Complexity	Heavily Modified a Pre-Existing Design

Software Challenges Faced

Issue	Solution
Faulty connection between ESP32 and Raspberry Pi	Rerouted bluetooth connectivity lanes to unassigned pins
Quirks in initial CSS styling	Established differentiation of object_ids and object_names; also began creating CSS groups
Icon and image import issues	Transitioned to local file paths and created custom icons to account for inconsistencies in default icons for different operating systems
Heartbeat sensor reading errors	Reread documentation and switched sensor from digital to analog output

Budget

Total Budget: $1,500



Progress and Plan For Completion

Present	1-2 Weeks out	3-4 Weeks out	5-6 Weeks out	9 Weeks out
PCBs mostly complete/Software in progress	Major Programing done/all PCBS assembled	Polish programs/ Housing made	Midterm	Showcase/Final

- Build the rest of the boards (Power, ESP-32 , and audio)
- Integrate mobile mode into the existing framework
- Finish by spring break!

Thank you!