E-P.I.P.

Environmental and Personal Information Processor

Alberto Rosas Cody Hum Dang Tran Noel Anilao

Table of Contents

I.
INTRODUCTION

Introductory Overview of *E-P.I.P.*

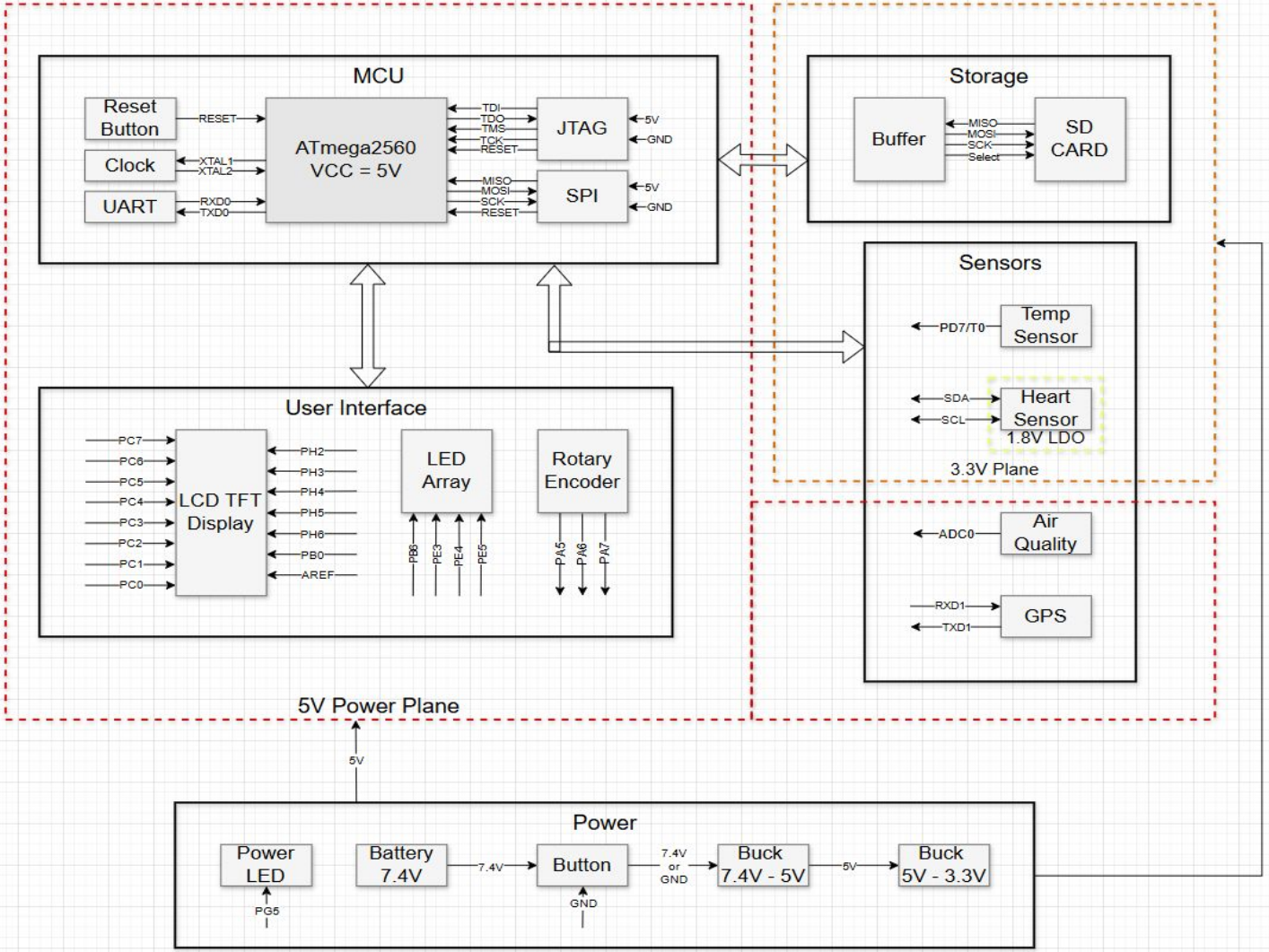
Main Purpose:

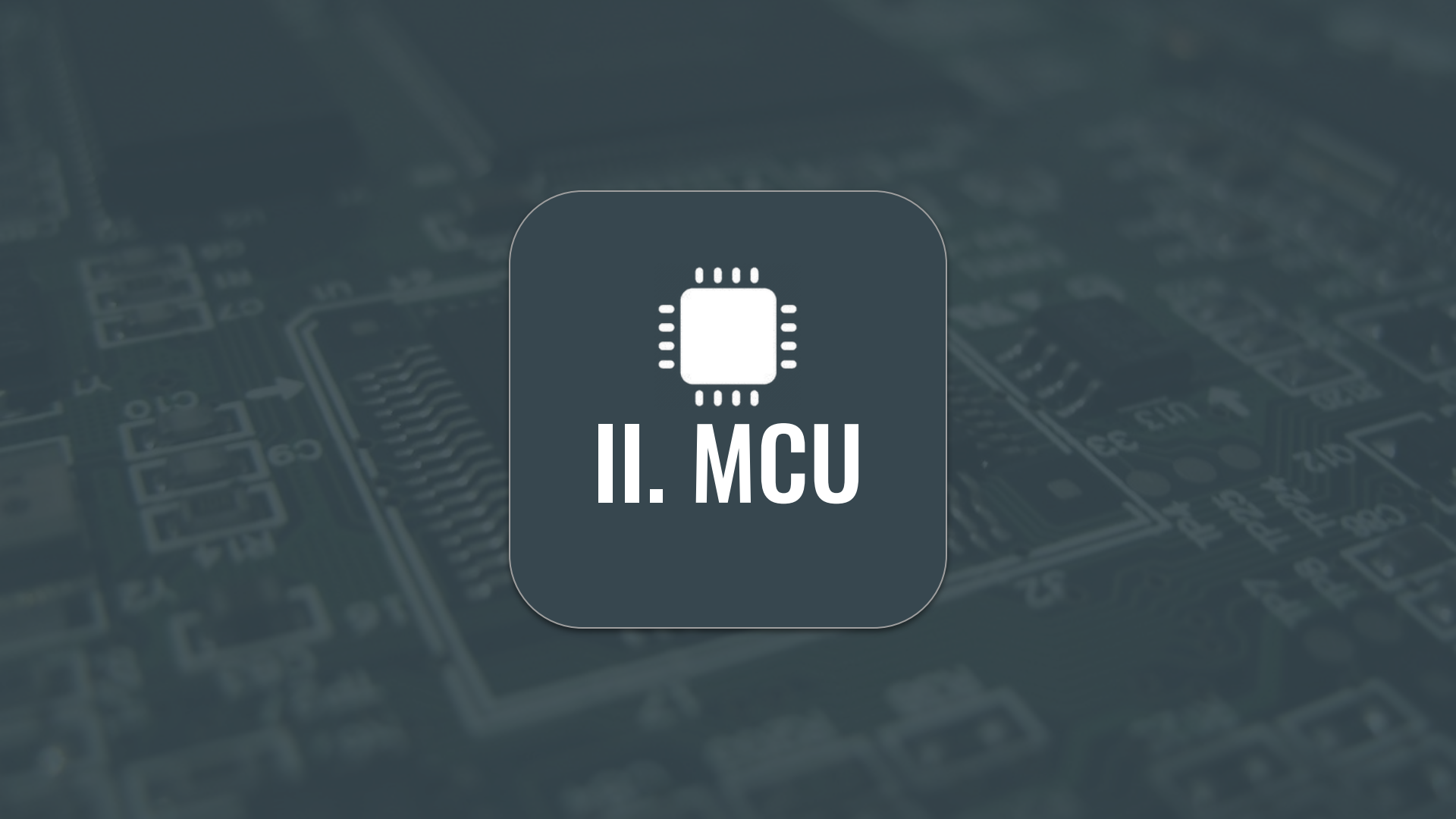
Safety device designed for harsh environmental conditions to keep users informed and safe by providing crucial data in critical situations.

Key Functions:

1. Temperature & Humidity
2. Air Quality
3. Location
4. Vitals
5. Store Data

E-P.I.P. Block Diagram




II. MCU

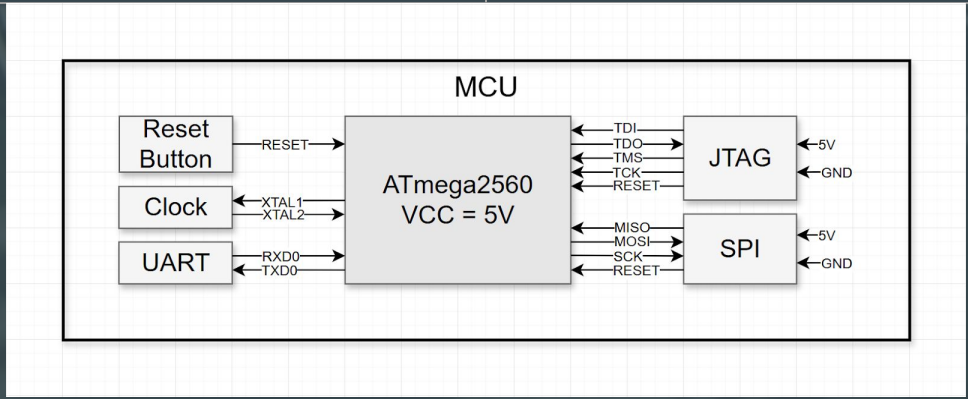
MCU

Purpose & Requirements:

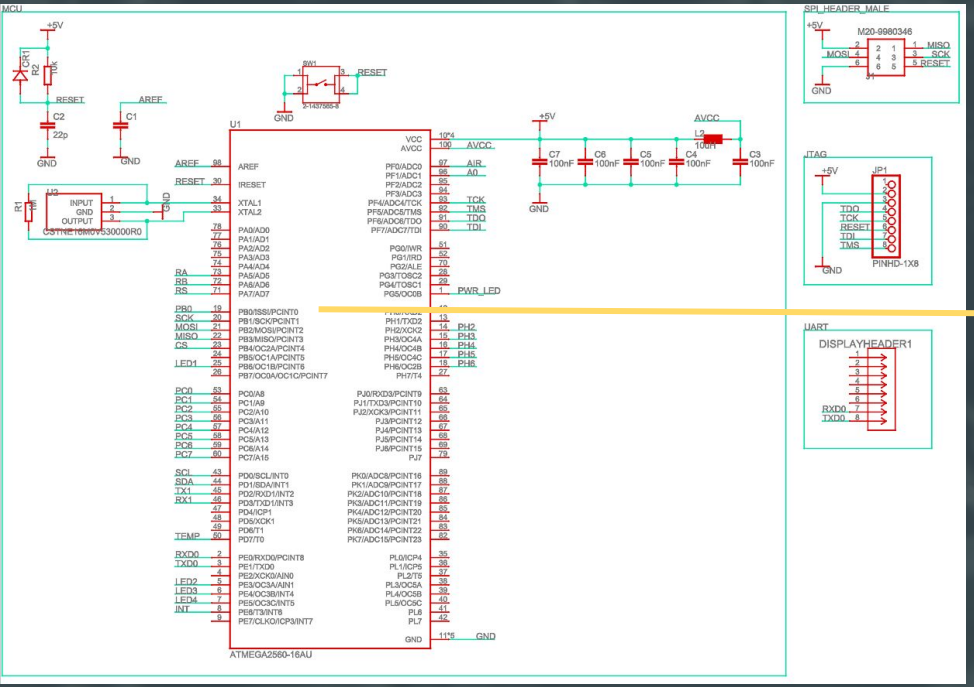
- Sensor data collection
- Input for User interface
- Display sensor readings

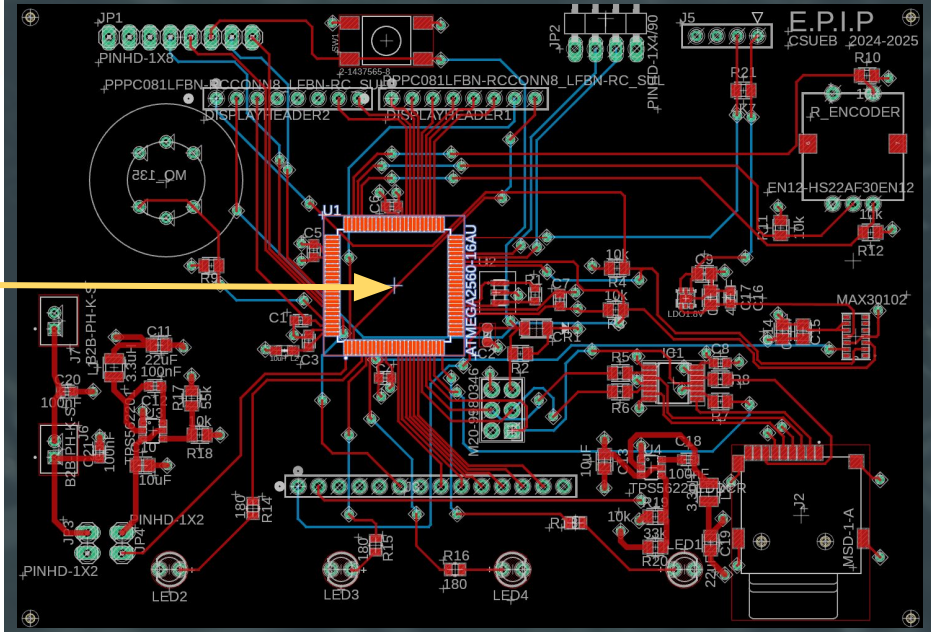
Part(s):

- ATmega2560
- 16MHz clock
- Reset Button
- JTAG, SPI, UART headers



MCU: Schematic & Layout



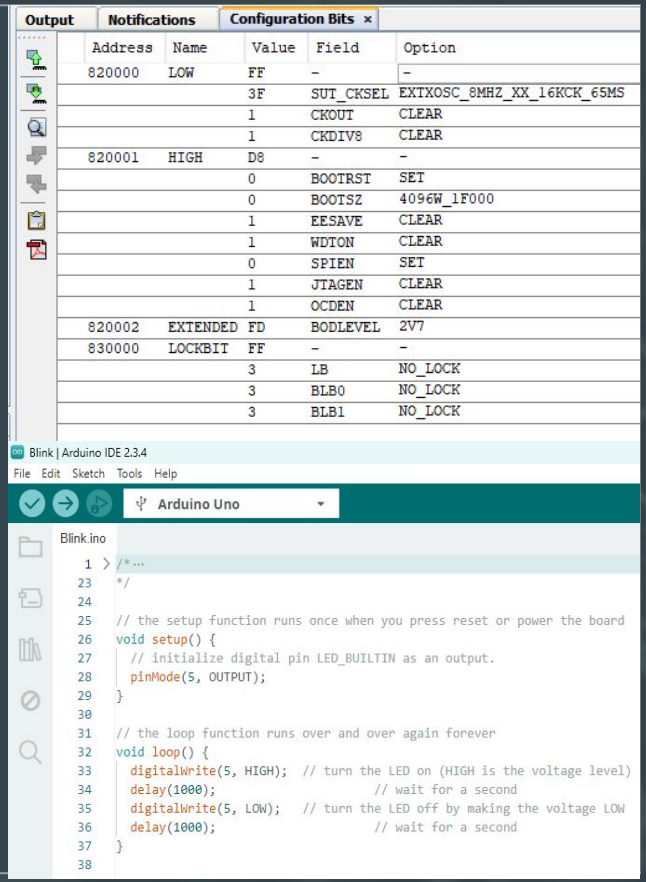


MCU: Programming

Bootloader compiled and exported from Arduino IDE as .hex file

Configuration bits set via MPLAB X IDE:

- LOW: 0xFF → Set external clock and No changes to timing
- HIGH: 0xD8 → SPIEN + BOOTRST, JTAG disabled
- EXTENDED: 0xFD → BODLEVEL = 2.7V
- Lock bits: 0xFF → No memory locking

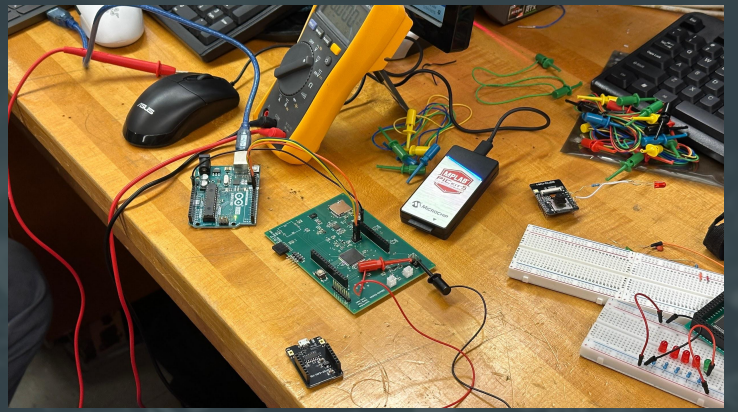


MCU: Milestones

Milestone 1: Verified stable power domains, but crystal oscillator failed { SPI programming unsuccessful }

Milestone 2: Bootloader successfully flashed via JTAG using MPLAB IPE and PICkit 5

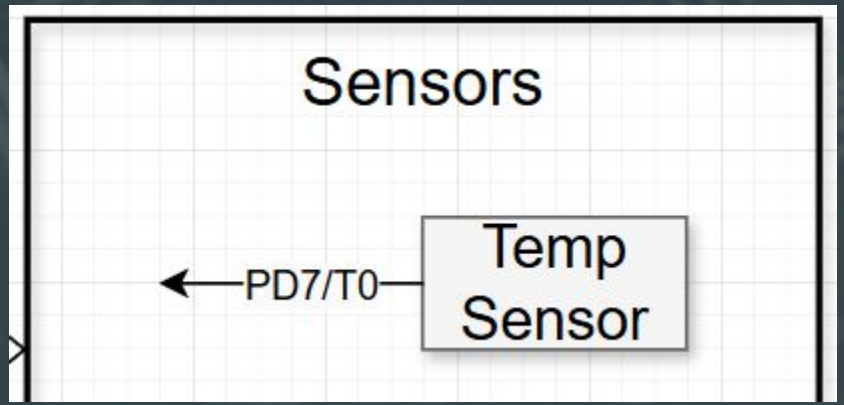
Milestone 3: Transitioned to UART uploads using a modified Arduino UNO as USB-to-Serial converter

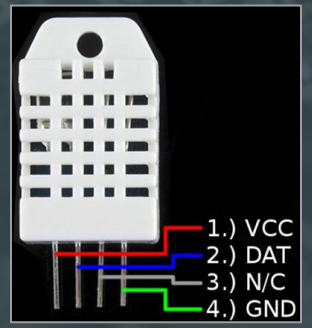


III. Temperature & Humidity

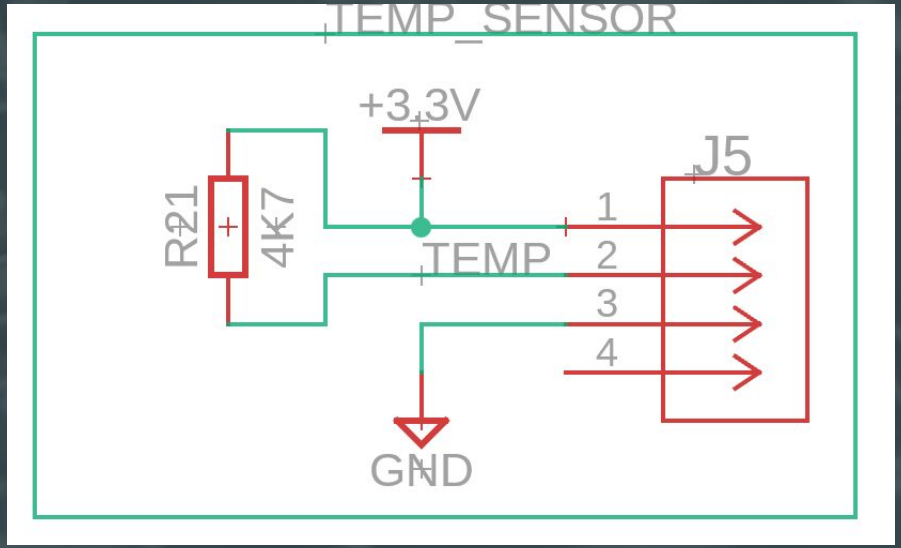
🌡️ Temperature/Humidity Sensor

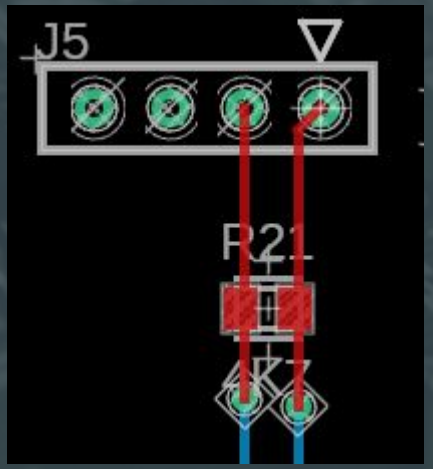
Requirements:	Software:	Integration:	Reach Goals:
• Read temperature • Read humidity • Store reading	• Sensor reading • Data storing	• Connect via female header pin • Top right of the pcb	• Read without 2 second delay





Temperature: Schematic & Layout





Temperature: Milestones

Milestone 1: No Problem, Turn on, and read temp/humidity

Milestone 2: Improvement

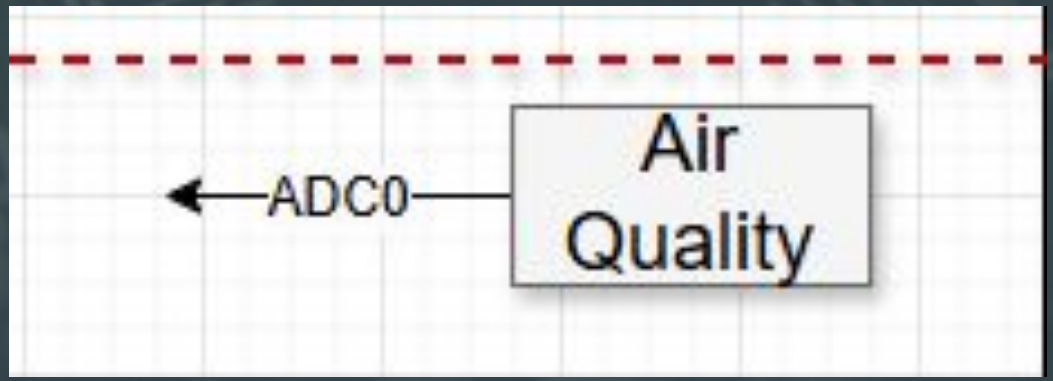
Made it read as fast as possible 2 second/ hardware issue

Met all core functional goals

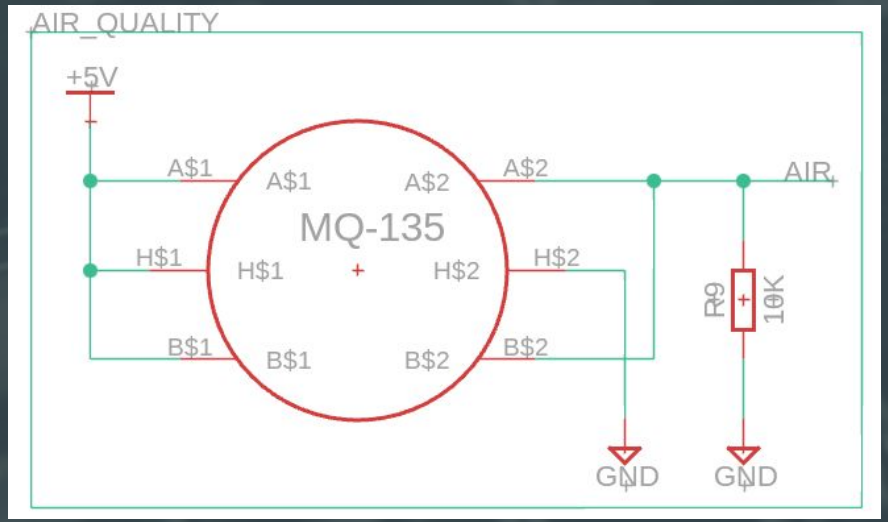
IV. Air Quality

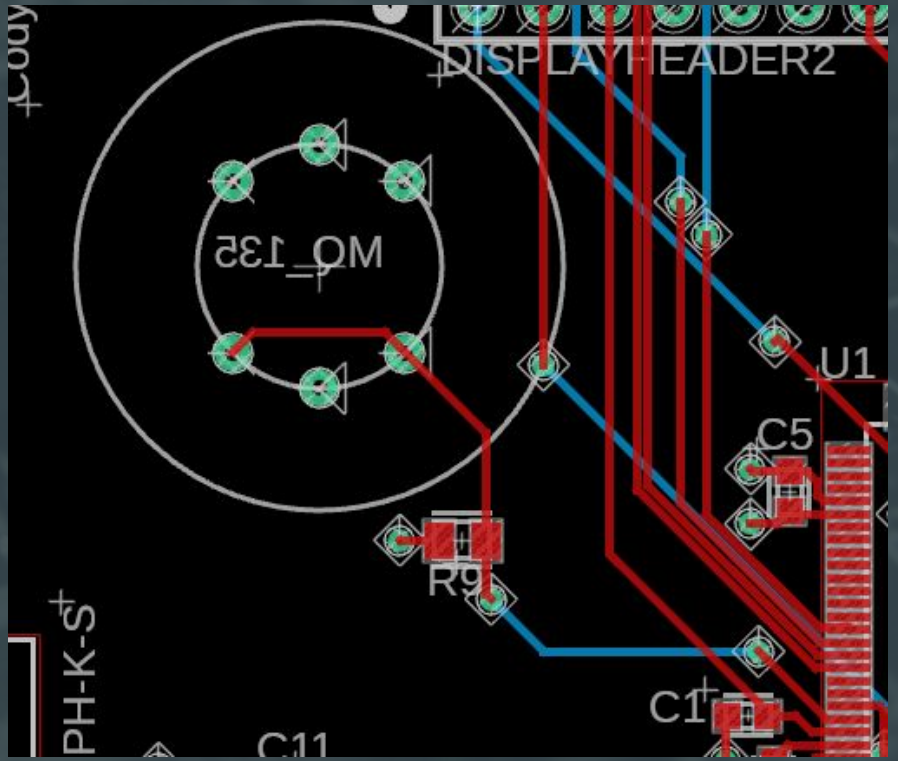
☁️ Air Quality: Overview

Requirements:	Software:	Integration:	Reach Goals:
• Read/Detect harmful gases (ppm) • Store readings	• Sensor reading • Data storing	• Connect into back of pcb • Soldered on	• Change Layout footprint



Air Quality: Schematic & Layout





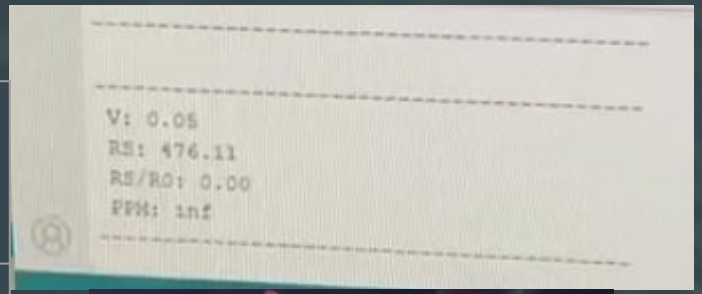
Air: Milestones

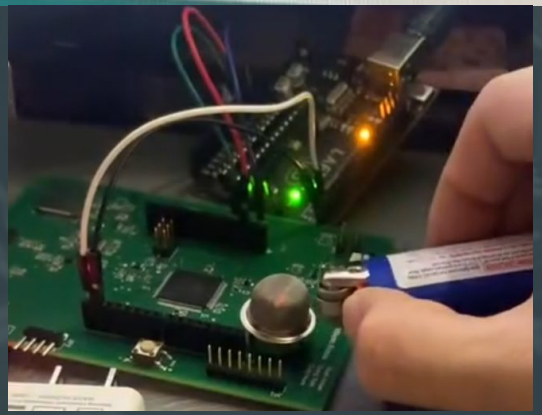
Milestone 1: Had to wire because of sizing

- Read air quality

Milestone 2: Shaved Sensor pins to solder onto PCB

- Read and read air quality
- Store them to be called on
- Fast effective readings





V. GPS

GPS: Overview

📝Requirements:

- NEO-6M GPS module w/ external antenna
- 5V input

</> Software:

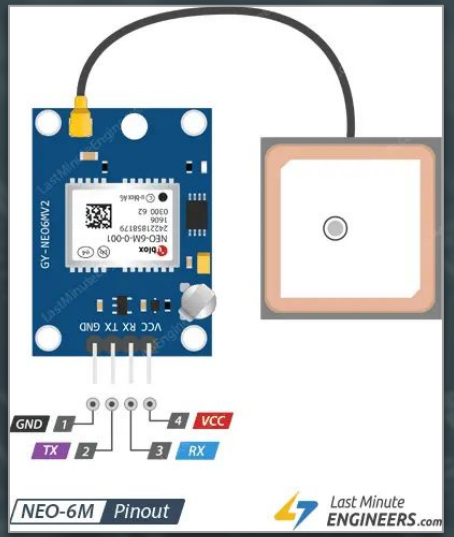
- Communication with MCU via UART
- TinyGPSPlus library
  - Locate real-time position
  - Find #'s of satellites
  - Get time

🔗Integration:

- Connection to MCU via RXD1 and TXD1

🚩 Reach Goals:

- Time sync without satellite coverage

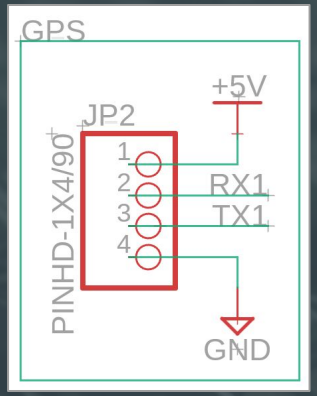


Dang T.

GPS: Schematic & Layout

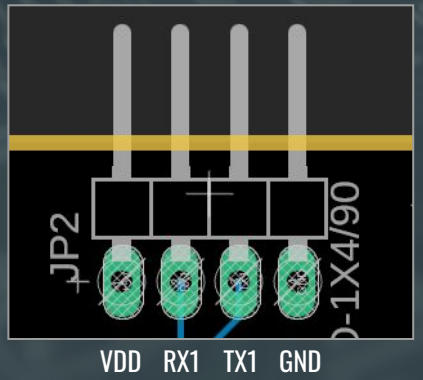
Schematic:

- Pin 1 (5V) connects to VDD
- Pin 2 (RX1) connects to RX (receiver) pin
- Pin 3 (TX1) connects to TX (transmitter) pin
- Pin 4 connects to GND



Layout:

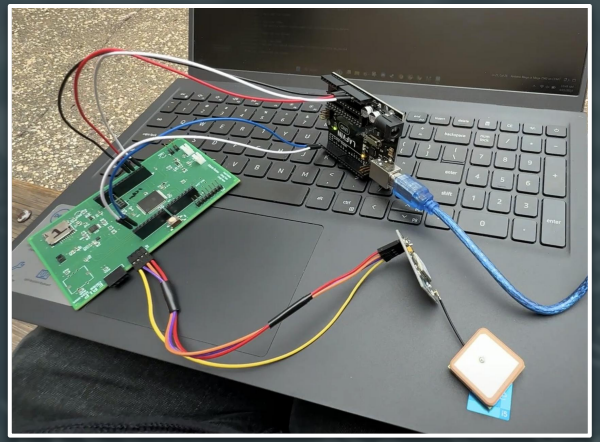
- Right angle 1x4 male pin header connects to the NEO-6M GPS module via female-to-female pin connection

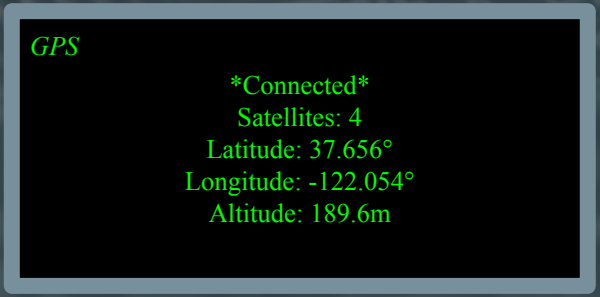


Dang T.

GPS: Milestones

1. Works with PCB board
2. Long initial cold start and limited satellite fixing
3. Continuous uptime of GPS module makes for ideal efficient operation
4. One limitation is the GPS antenna sensitivity
   - requires user to be outdoors or in open space to maintain reliable connection



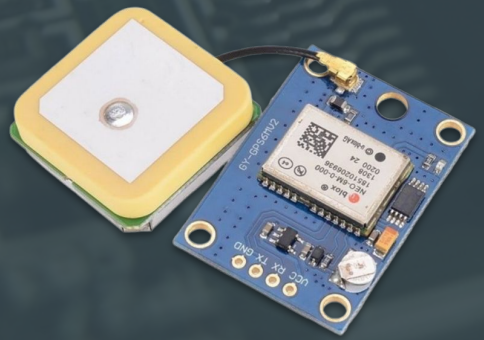


GPS Display Example

GPS: Summary

- NEO-6M module is key to providing accurate and reliable satellite-based location tracking;
- NEO-6M has its limitations, such as long initial start time & limited satellite acquisition;

- We use TinyGPSPlus library to properly integrate GPS functionality;
- Communication between NEO-6M and MCU is handled via UART;
- *Overall, our implementation aims to deliver a practical and dependable method for real-time location tracking*



GPS

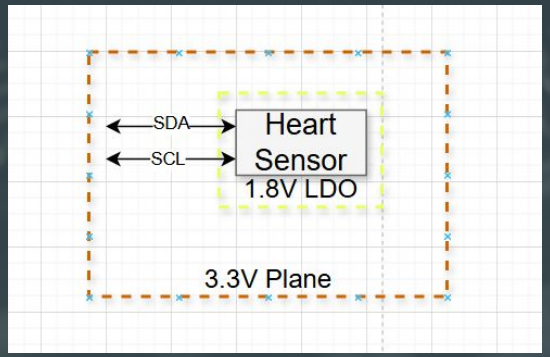
Connected
Satellites: 4
Latitude: 37.656°
Longitude: -122.054°
Altitude: 189.6m

Dang T.

VI. Vitals

Vitals Overview

📝 Requirements:	</> Software:	🔗 Integration:	🚩 Reach Goals:
• 1.8V LDO powering sensor • 3.3V powering LEDs within module	• Arduino Software • SparkFun Particle Sensor Arduino Library	• Connection to MCU via I²C	• Faster sensor readings • Cleaner sensor output



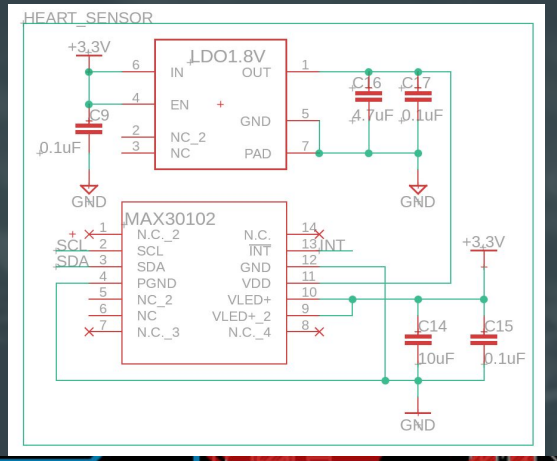
Vitals Schematic & Layout

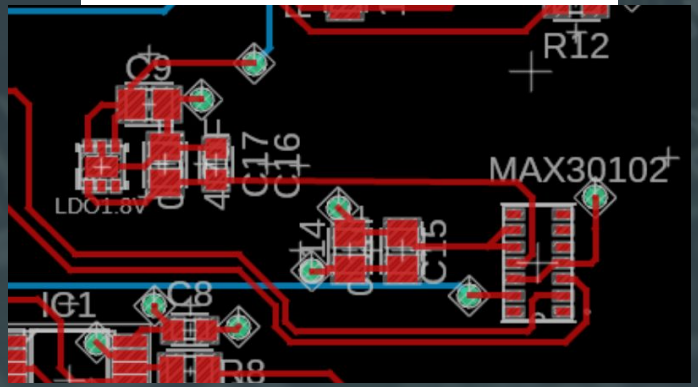
Schematic:

- Pin 2 & 3 Connects to MCU via I²C
- Pin 9 & 10 3.3V domain
- Pin 11 1.8V LDO
- Pin 12 GND

Layout:

- Center-Right of PCB
- Decoupling Capacitors on LDO and VLED



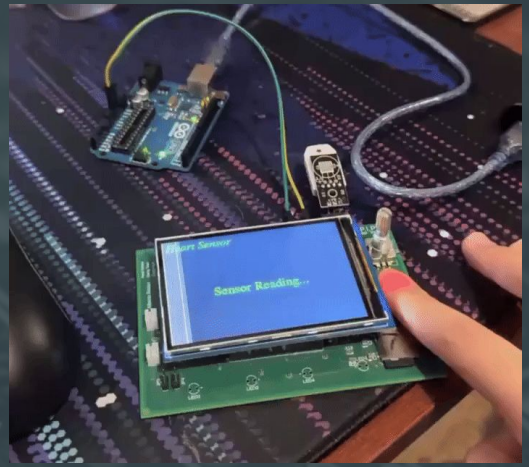


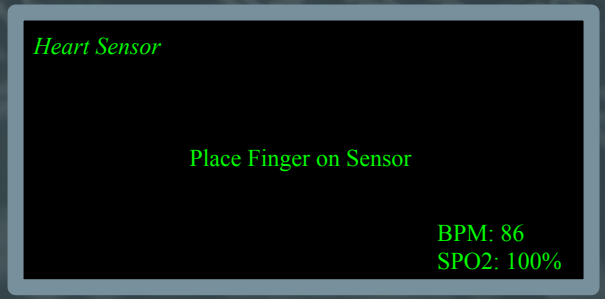
Vitals Milestones

Milestone 1: Verified all power domains (3.3V & 1.8V) and working LED control through I2C

Milestone 2: Sensor readings achieved allowing for heart rate and blood oxygen calculations

Milestone 3: BPM and SPO2 code implementation and fine tuning for better accuracy



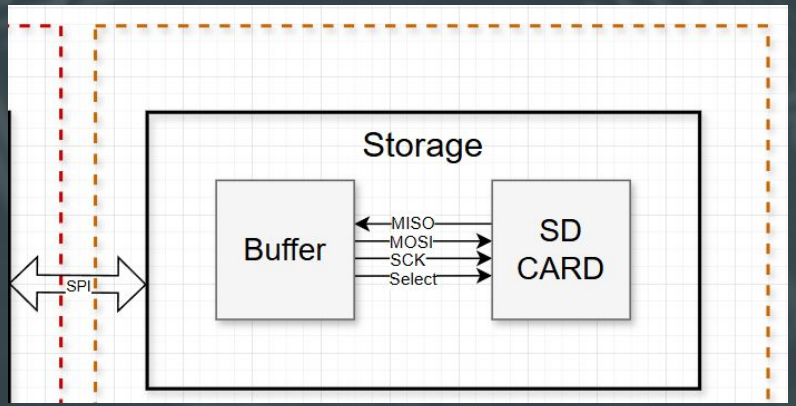


Vitals Display Example

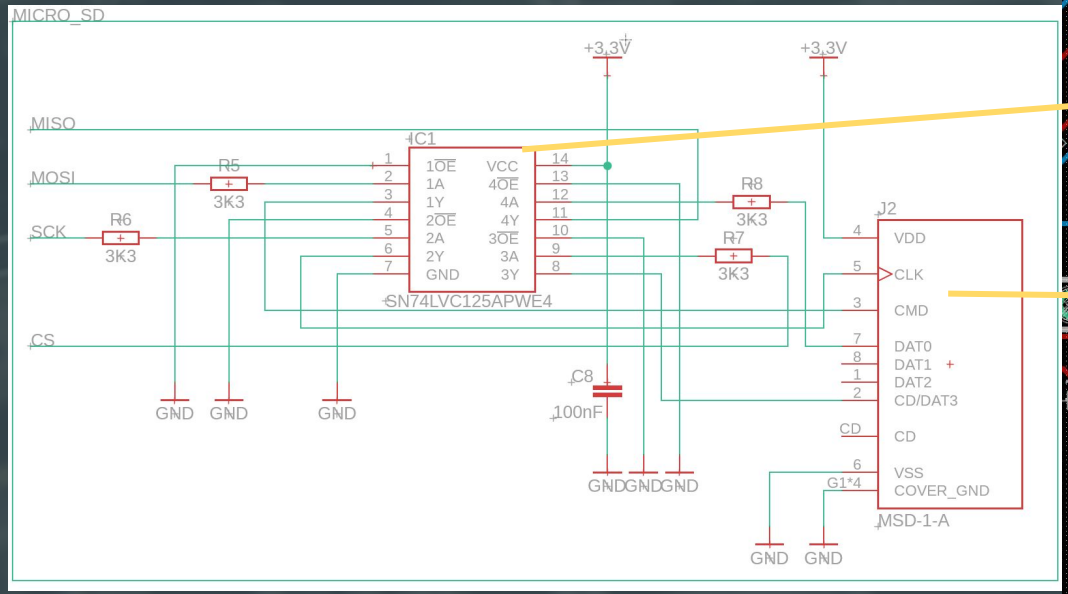
VII. Storage

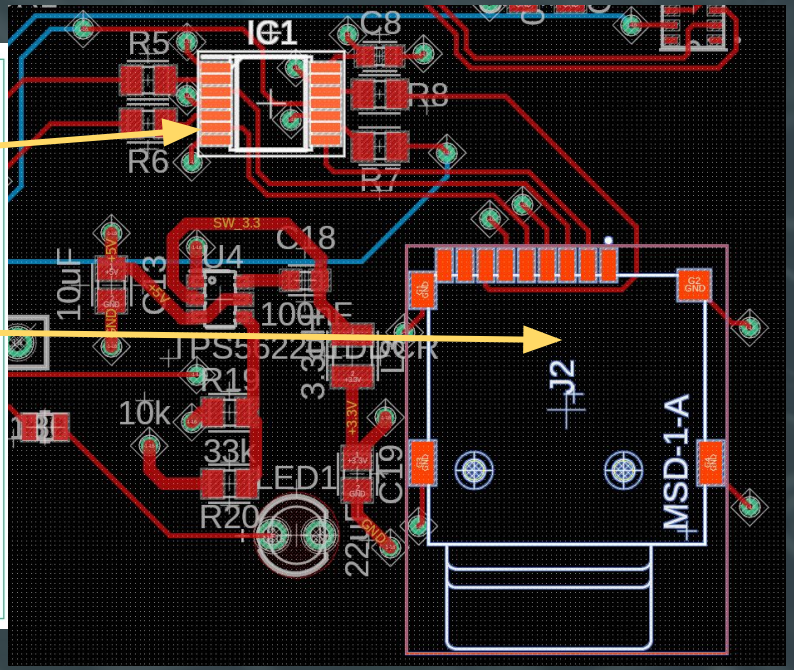
💾 Storage: Overview

Requirements:	Software:	Integration:	Reach Goals:
• Log sensor alerts • Timestamp entries • Store user data	• Event-driven logging • Filename versioning	• Connect via SPI to MCU • Used sensor flags for events • Sync with GPS/Uptime	• On-screen log viewer • Program to format Storage and add user file



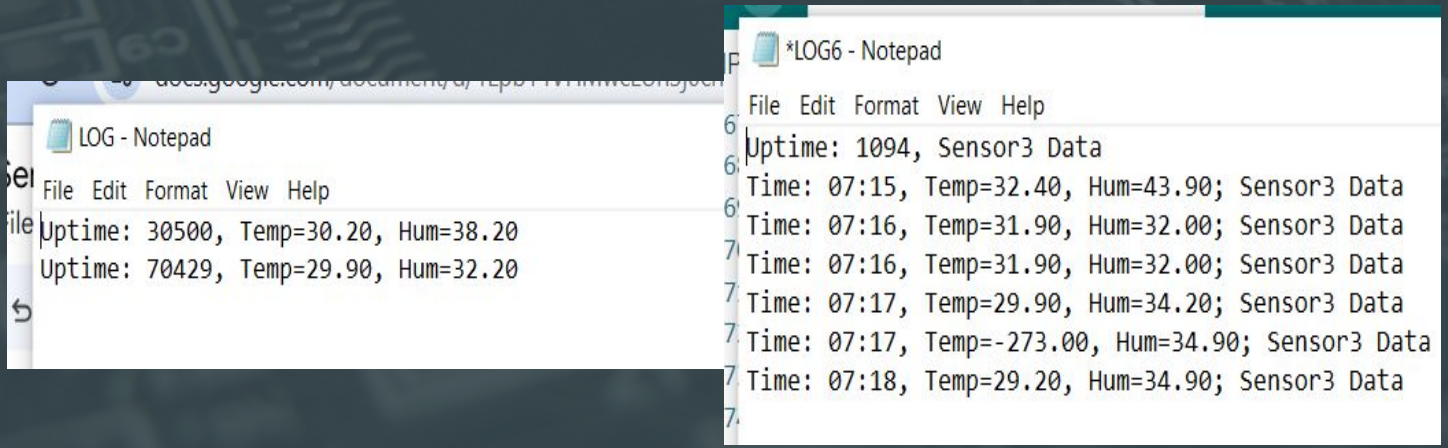
Storage: Schematic & Layout





Storage: Milestones

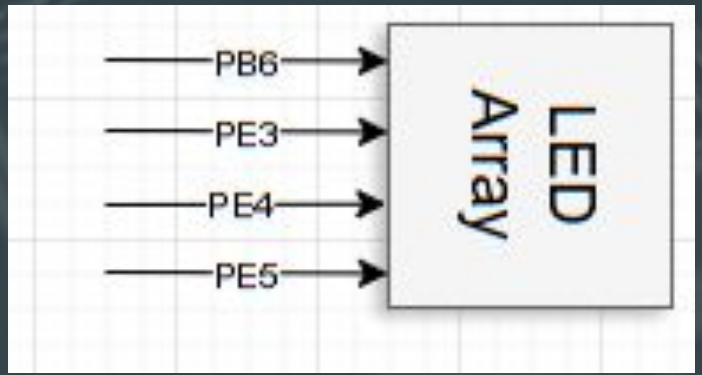
Milestone 1: Power Verification & Card Detection
Milestone 2: Logging Functionality and session numbering



VIII.
Rotary Encoder & LED Array

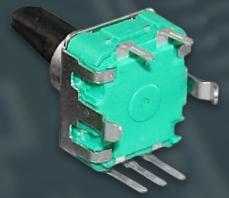
🚨 LED Array: Overview

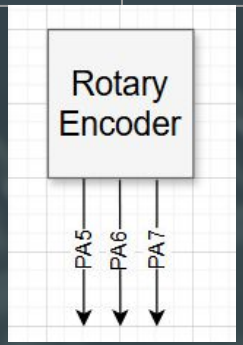
Requirements:	Software:	Integration:	Reach Goals:
• Light up • Read stored data • Detect flag	• Check store data is flagged • Light up if over limit	• Soldered on Red LED to GPIO lines	• Integrate light fade to replicate blinking



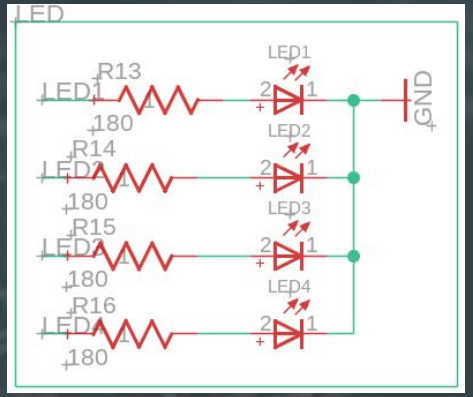
⌨ Rotary Encoder: Overview

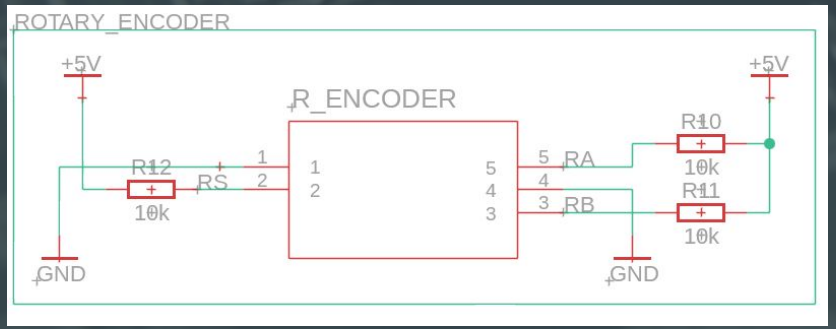
Requirements:	Software:	Integration:	Reach Goals:
• 5V power domain • Switch multiple menus on display • User input	• State machine • Debounce delay • Idle state + Rotation modes	• MCU pins • 27 Button press • 28 -Rotation • 29 +Rotation	• Less false triggers

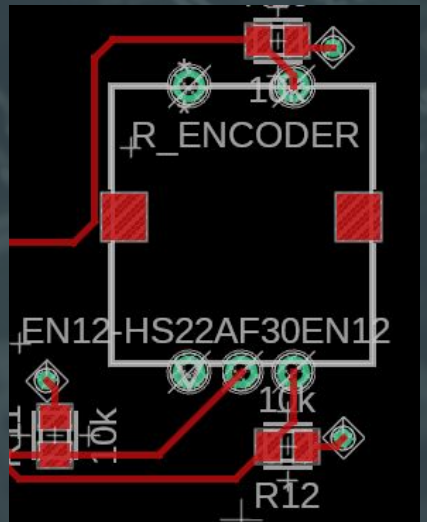




Rotary Encoder & LED Array: Schematic & Layout





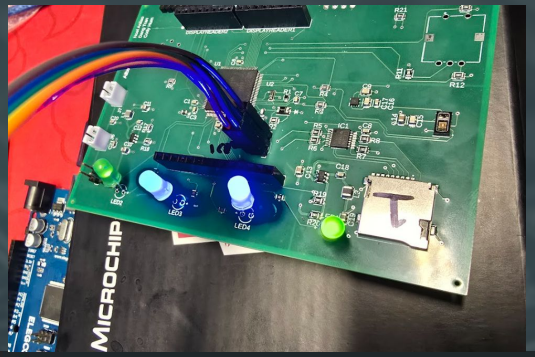


Rotary Encoder & LED Array: Milestones

Milestone 1: Power verification correct for both Rotary Encoder & LED Array, however input from RE was incorrect (CLK & GND).

- Corrected in software via GPIO mapping

Milestone 2: Mechanical bounce/Signal stability refined for Rotary Encoder. LED Array implementation for respective sensor display menus



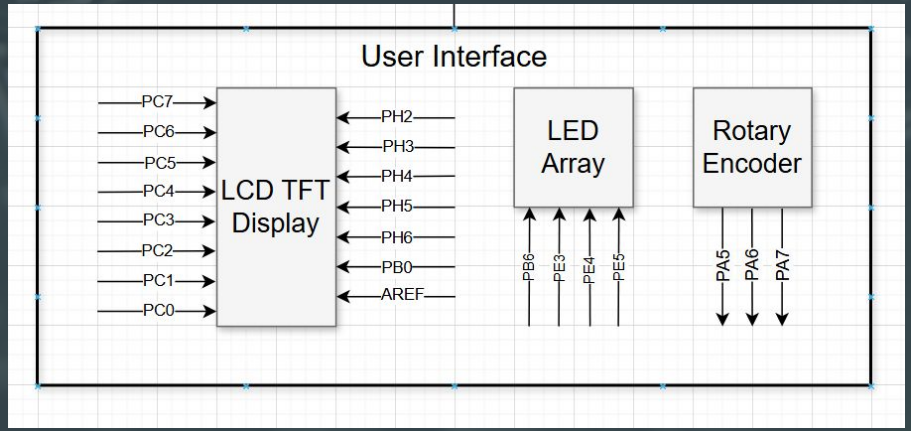

```
void input() {
  int currentStateCLK = digitalRead(ENCODER_CLK);
  // Check if the rotary encoder has turned
  if (currentStateCLK != lastStateCLK && currentStateCLK == HIGH && counter!=-1) {
    counter = (digitalRead(ENCODER_DT) != currentStateCLK) ? counter+1 : counter-1;

    // Loop counter from 0 to 3
    if (counter > 3) counter = 0;
    if (counter < 0) counter = 3;
  }
  // Check if the button is pressed
  if (digitalRead(ENCODER_SW) == LOW) {
    counter = (counter == -1) ? counter = 0 : counter = -1;
    delay(200);  // Simple debounce
  }
  lastStateCLK = currentStateCLK;
}
```

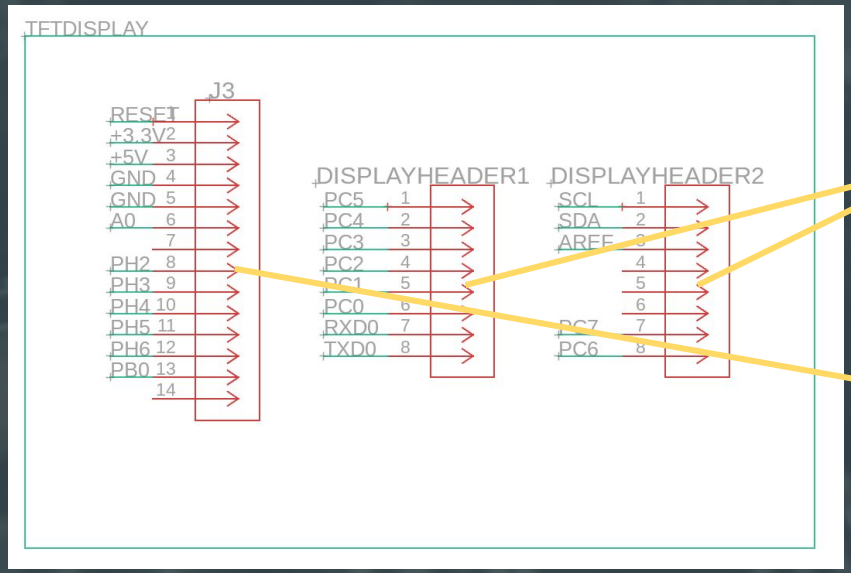
IX. Display

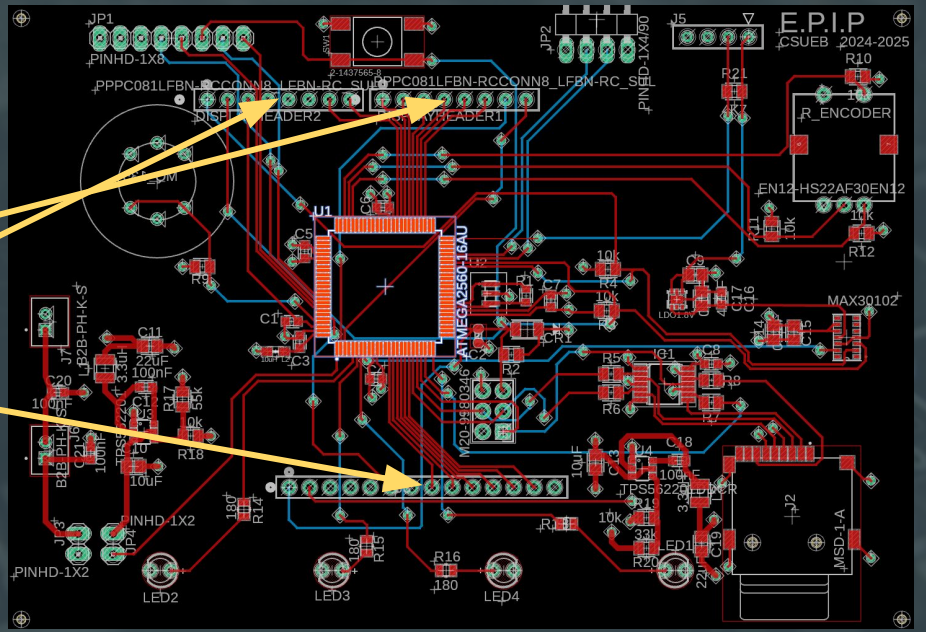
Display Overview

📝 Requirements:	</> Software:	🔗 Integration:	🚩 Reach Goals:
• Communication via GPIO headers on PCB • 5V power	• Adafruit_GFX library • MCUFRIEND_kbv library	• GPIO pin headers connected to the MCU (Parallel Interface)	• Faster GUI navigation • Fancier/cleaner GUI



Display Schematic & Layout



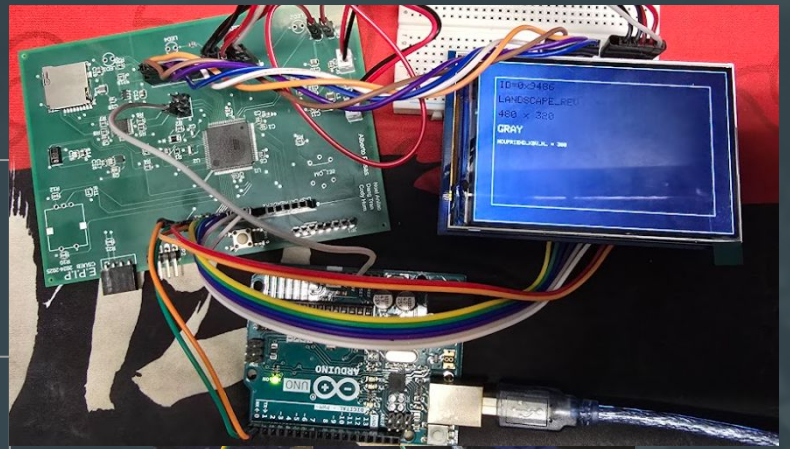


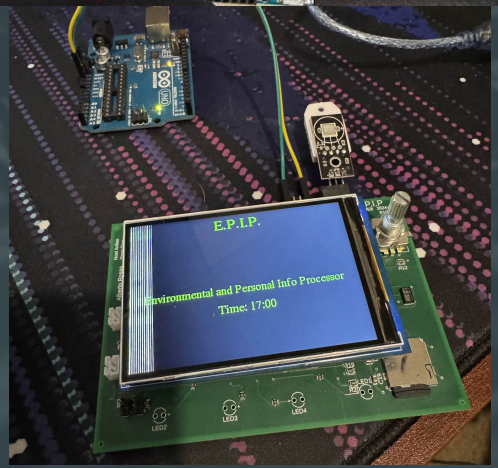
Display Milestones

Milestone 1: Verified Power domains, and correct display output via parallel pins

Milestone 2: Sensor integration and respective display menu outputs with data

Milestone 3: GUI refinement





X. Firmware

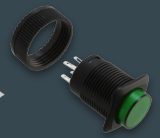
Startup: Initializes all sensors, LCD, and user interface, power-on sequence with green LED confirmation.
Sensor Management: Executes read functions for temperature/humidity, air quality, and GPS sensor every 3 sec in a non blocking loop. Vitals sensor is only read when user prompts it.
User Interface: Rotary encoder for screen navigation, Update screen by redrawing or updating data, LED alerts are triggered by relevant sensor Flags
Data Logging: Logs environmental and vital data to Micro SD card.


⚡
XI. Power Supply

⚡ Power Supply: Block Diagram

📝 Requirements:

- x2 3.7V lithium-ion batteries
- x2 buck-switching regulators
- Switch pushbutton →

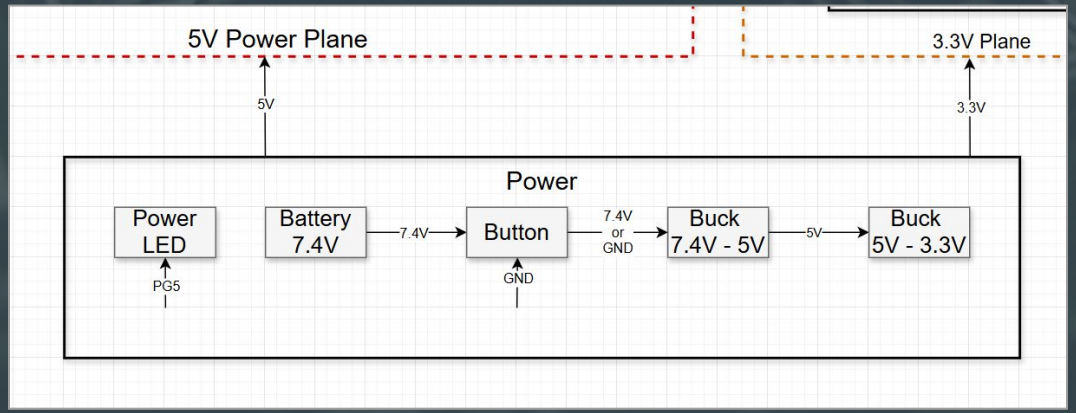


🔗 Integration:

- Portable power supply
- Regulated via buck-switching regulators
- 5V and 3.3V domains

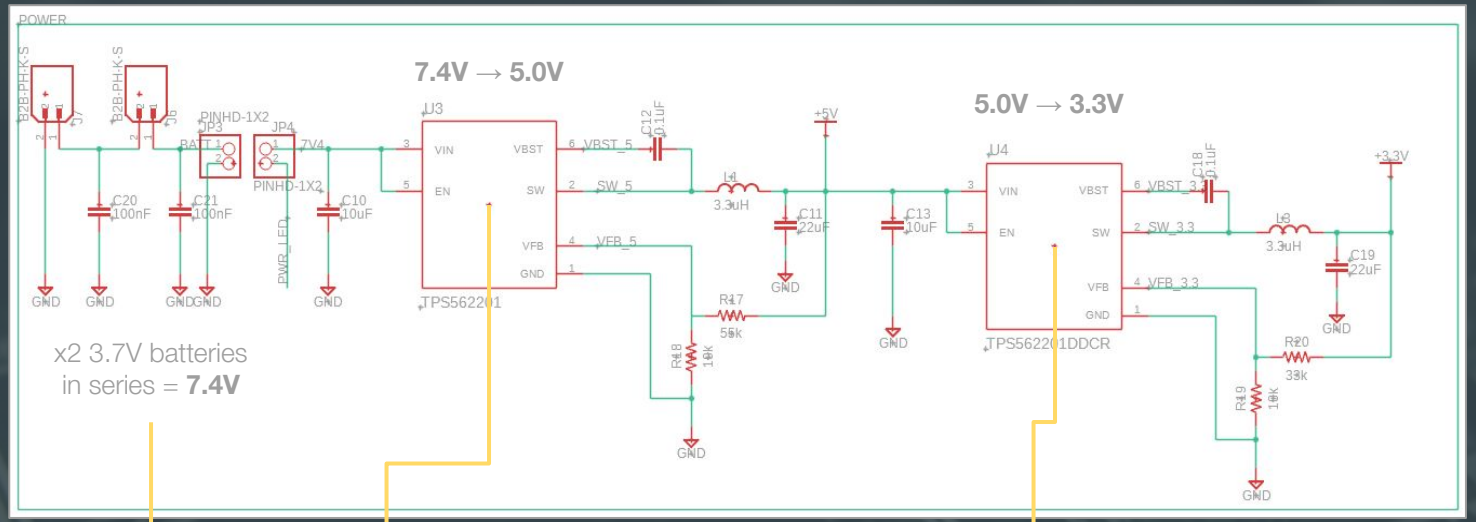
🚩 Reach Goals:

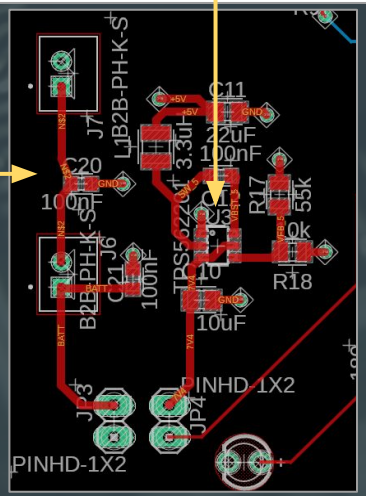
- Eliminate gradual voltage dissipation

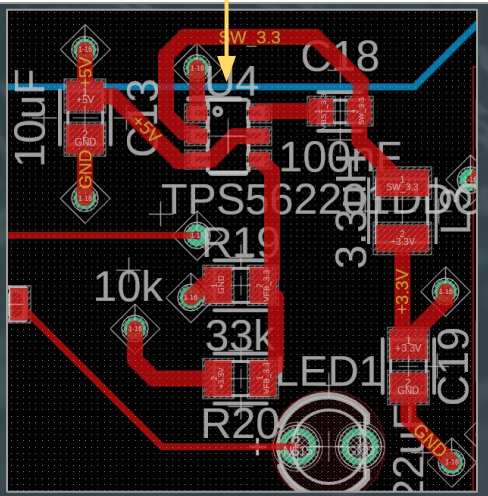


Dang T.

⚡ Power Supply: Schematic & Layout







Dang T.

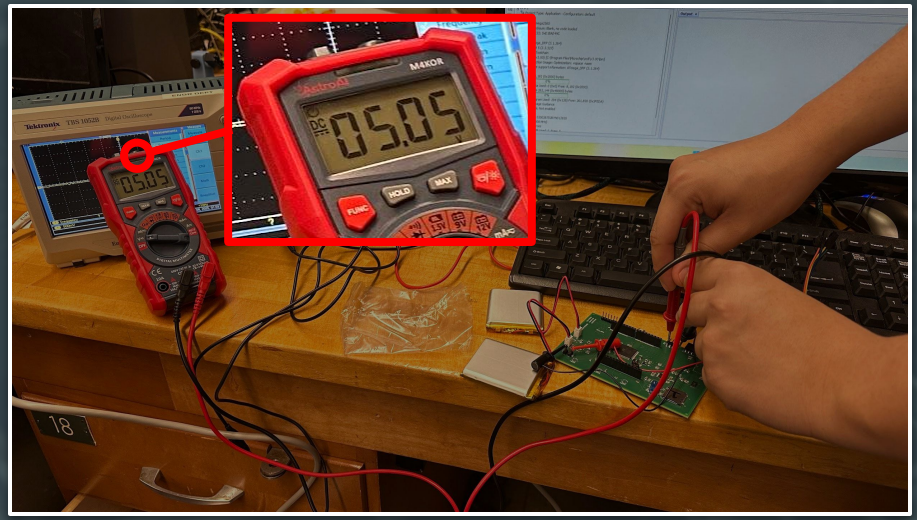
⚡ Power Supply

Milestones

1. Successful confirmation that the buck regulators can step down to 5V and 3.3V
2. Minor residual voltage after powering down; gradually dissipation to 0.001V

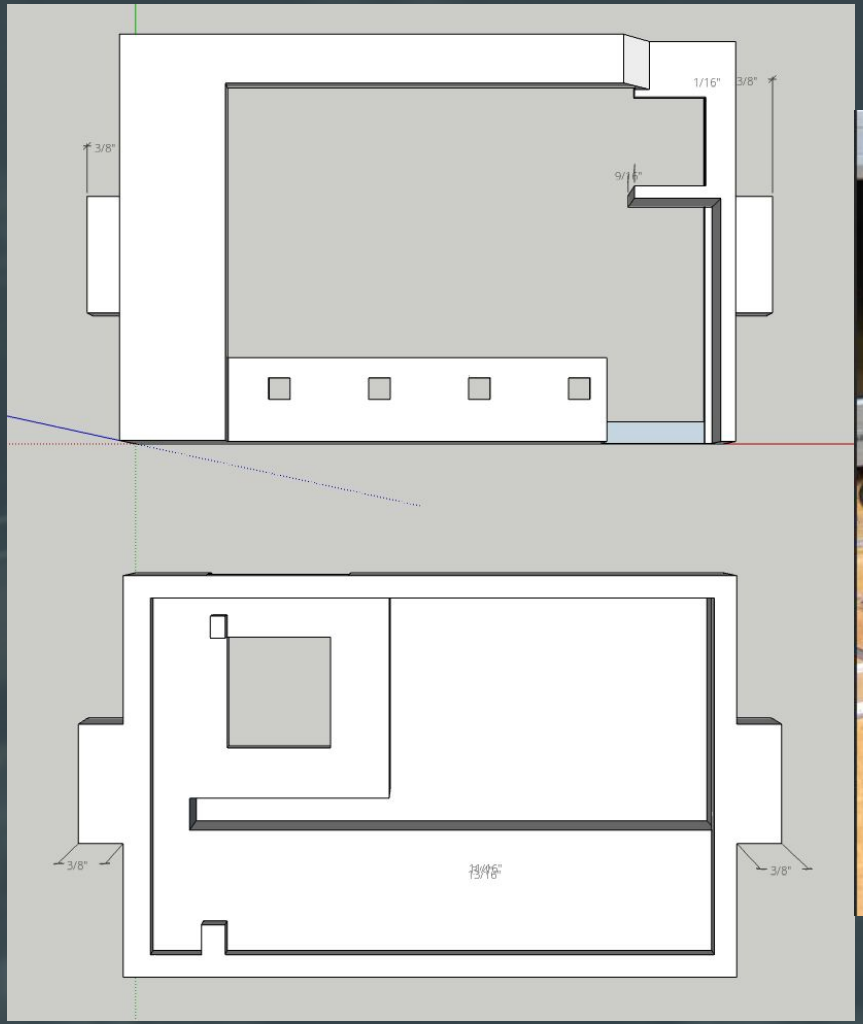
Summary

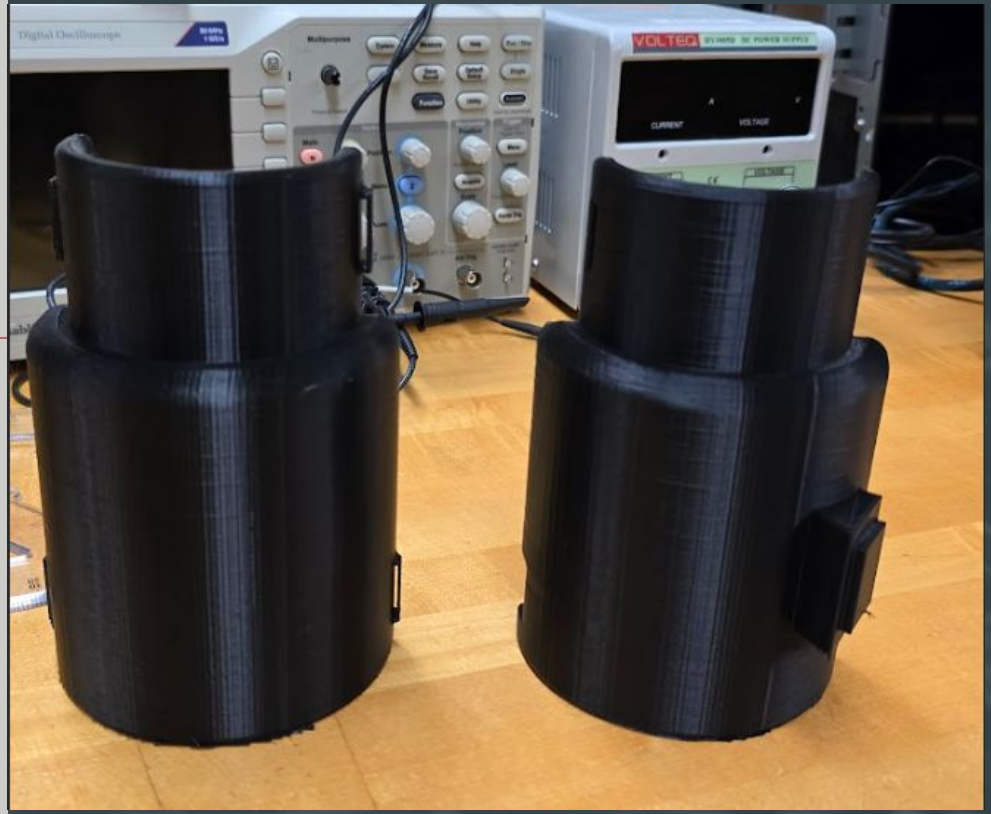
- Operates at 7.4V (x2 3.7V lithium-ion batteries in series)
- Two voltage domains, both regulated using buck-switching regulators
- Tiny issue in our layout design; not as efficient, but it works



Dang T.

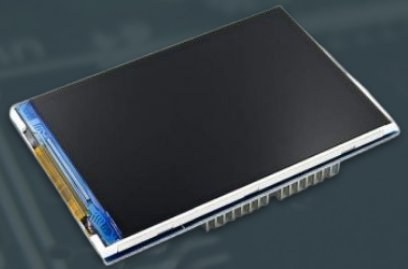
XII.
CASING

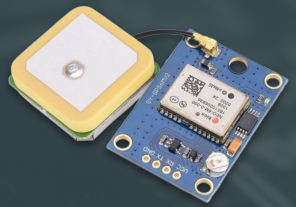

1/16"
3/8"
3/8"
9/16"
3/8"
3/8"


VOLTEQ
CURRENT
VOLTAGE

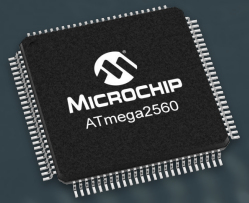
XIII. CONCLUSION

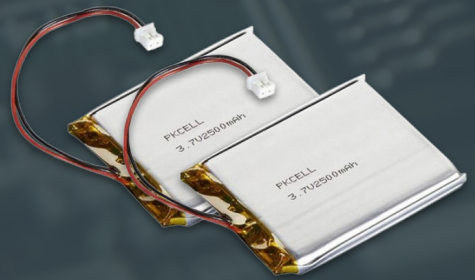
Conclusion

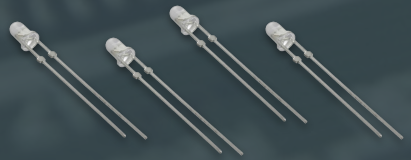




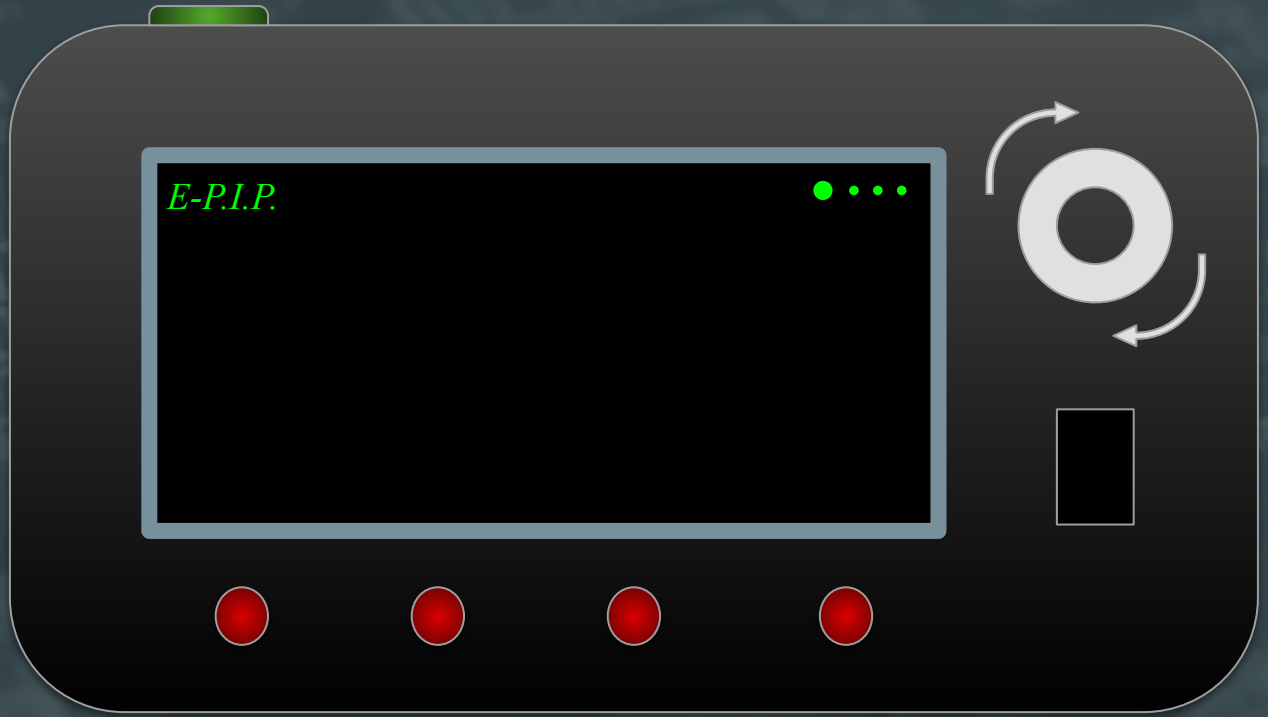

MQ135


MICROCHIP
ATmega2560

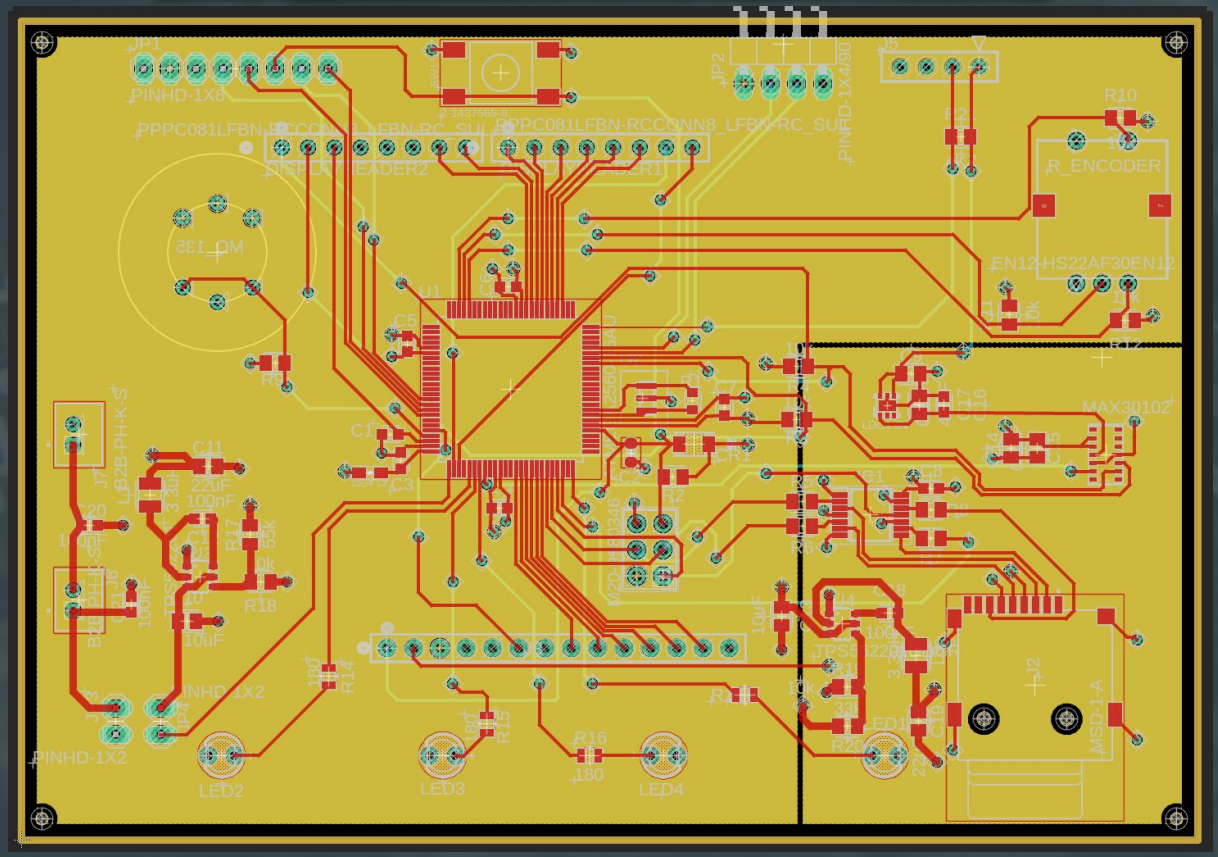

PKCELL
3.7V2500mAh
PKCELL
3.7V2500mAh



Conclusion



Q&A


PINHD-1X8
PPPC081LFBN-RCCONN8_LFBN-RC_SUI
R_ENCODER
EN12-HS22AF30EN12
MAX30102
MSD-1-A
LED2
LED3
LED4