**A STEP-BY-STEP GUIDE – KICAD**

CH1. **Introduction:**

Main tools to cover for a successful BB:

1. Schematic – Layout/Circuit Design (EESCHEMA)
2. Symbol Editor – Edit symbol/Create new symbols (PCBNEW)
3. PCB Editor – do layout design, arrange and create board design for the circuit
4. Gerber viewer – Check out exported gerber files
5. Plugin – add extensions and other tools.

Workflow of PCB Design.

A white paper with writing on it

AI-generated content may be incorrect.

PCB material: FR4 flame-retardant glass-reinforced epoxy laminate material used to make printed circuit boards (PCBs). It's a composite material made of woven fiberglass cloth and a flame-retardant epoxy resin binder

* Excellent Isolator
* Non- absorbent of water
* Operates in High temperatures
* Paper based materials
* Aluminum

<https://github.com/kicad> - Regularly updates symbols, and footprints

<https://github.com/Digi-Key/digikey-kicad-library> - Library for Digikey tools.

PROJECT :- MCU datalogger with built in 512K EEEPROM.

**Intro :** A microcontroller (MCU) based data logger with 512 KB EEPROM is a system that reads and stores data (such as temperature, pressure, sensor readings, etc.) into an electrically erasable programmable read-only memory (EEPROM) for later retrieval.

1. Components Involved

1. Microcontroller (MCU):  
   – Core processing unit that reads sensor data and controls the flow.  
   – Examples: ATmega328P (Arduino), STM32, ESP32, etc.
2. Sensors:  
   – Devices connected to the MCU to measure data like temperature, humidity, voltage, etc.
3. EEPROM (512 KB):  
   – Non-volatile memory where data is stored.  
   – 512 KB = 524,288 bytes.  
   – Example ICs: 24LC512 (64 KB), AT24CM02 (256 KB), or combine multiple via I²C or SPI.
4. Real-Time Clock (RTC) (optional):  
   – For timestamping data (e.g., DS3231 module).
5. Power Supply:  
   – Battery or DC input to run the MCU.
6. Communication Interface (optional):  
   – USB, UART, SD card, or wireless module to retrieve data.

A diagram of a temperature sensor

AI-generated content may be incorrect.

2. System Operation Flow

1. Initialization:
   1. MCU sets up I²C/SPI communication with EEPROM and RTC.
   2. Initializes all sensor peripherals.
2. Data Acquisition Loop:
   1. At regular intervals (e.g., every 10 seconds), the MCU:  
      ▸ Reads data from sensors.  
      ▸ Gets timestamp from RTC (if used).  
      ▸ Formats the data (e.g., CSV or binary).  
      ▸ Writes data to EEPROM sequentially.
3. EEPROM Management:
   1. Keeps a write pointer to know where to write next.
   2. Can implement circular buffer (overwrite oldest data when full).
4. Data Retrieval:
   1. Data can be dumped to PC via UART, USB, or wireless (e.g., ESP32).
   2. Optionally triggered by a command or switch.

3. Data Storage Planning

Let’s say you log 2 sensors + timestamp:

* Temperature: 2 bytes
* Humidity: 2 bytes
* Timestamp: 4 bytes  
  → Total per log = 8 bytes

With 512 KB = 524,288 bytes, you can store:

524,288 ÷ 8 = 65,536 records If logging every 10 seconds:  
→ 65,536 × 10 sec = ~182 hours ≈ 7.6 days of continuous logging

**NEW PROJECT CREATE AND PUSH TO GIT & GITHUB:**

Store project history, experimental branches in a repo, merge or discard branches.

Git bash in the project folder

Git init

Git status -> to check status

Git add . (add for single file // add . for all files in directry)

vim .gitignore 🡪 add ignore files, esc, :wq save and quit

**BACKUPS AND GERBERS**

dir – display all files in the directry

git branch (check for branches

git log (logs )

git commit -m

after every edit commit the changes to github.

git remote add origin https://github.com/Ruthvik-reddy-A/PCB--MCU-Datalogger.git

git branch -M main

git push -u origin main

git pull origin *branchname*

*To experiment : to create new branches* .

git checkout -b experimental (exp name)

git merge -> if branch would like to add changes

git branch -d *name* - > del branch

-D override delete

CH2. **PCB Design Part 1 – EESCHEMA :**

COMPONENTS USED:

|  |  |
| --- | --- |
| Capacitor | 5 |
| Resistor | 7 |
| [I2C Serial EEPROM(*24LC1025*](https://ww1.microchip.com/downloads/aemDocuments/documents/MPD/ProductDocuments/DataSheets/24AA1025-24LC1025-24FC1025-1024-Kbit-I2C-Serial-EEPROM-20001941M.pdf)*)* for data storage for logging data using i2c (512kB) | 2 |
| Led | 2 |
| Crystall oscillator (https://www.electrical4u.com/crystal-oscillator/) | 2 |
| Mounting holes | 4 |
| [Atmega328P-AU (8-bit Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash)](https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf) | 1 |
| [DS1337S (I2C Serial Real-Time Clock)](https://www.analog.com/media/en/technical-documentation/data-sheets/DS1337-DS1337C.pdf) – real time clock with two programmable timeofday, clock/calender provides s,m,h,day,date,month,year(1.8-5.5V) | 1 |
| Battery | 1 |

|  |  |
| --- | --- |
| Connectors: |  |
| 4-pin (I2C & SERIAL UART) | 2 |
| 9-PIN (GPOI) | 1 |
| 2x3 PIN (In Circuit Serial Programming) | 1 |

Once w know the component s used we open schematics and select place symbols tab and search for components to place.

We split up connectors to place in another sheet of schematic using *draw heirachial sheets)*

First components are placed like this before connecting them.

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

Association:

In this step we edit values of the component and add values

We open Assign footprints from tools section and assign the footprints for each component

We prefer to take SMD’s for cap, res, ind and led’s.

If any new downloaded comp are there then adding those footprints to library before assigning.

Wring:

During wiring we add VCC and GND labels to the schematic an also include a power flag(*to inform the Electrical Rules Check (ERC) about the nature of a net and to avoid errors during the ERC process)*

Labels and Nets:

We connect labels and name the pins of all the components we want to use. Connect the conectors to the main schematics.

ERC: Check and solve any electrical rule checks. Add comments to the schematics

A screenshot of a computer

AI-generated content may be incorrect.

Now we set up another GitHub branch to split 2-layer and 4-layer PCB.

**Command and name: git checkout -b 2layer**

CH3. **PCB Design Part 2 – LAYOUT :**

**A screenshot of a computer

AI-generated content may be incorrect.**

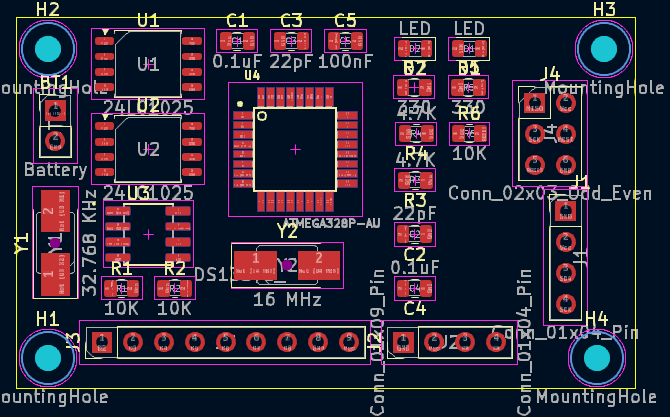
Setup, Outline and Constraints: Board Setup and figuring out the size of PCB

Place components:

We arrange the components wrt to the schematic wiring, first place the large components and then adjust the smaller ones in the gap.

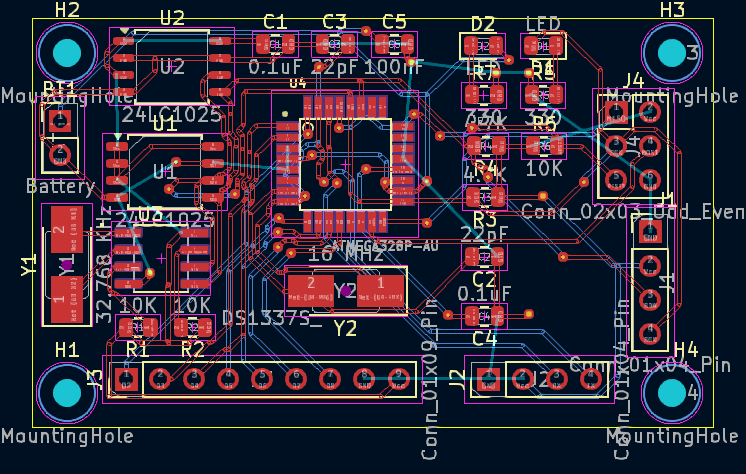
Degree the size if you can decrease the board map. Keep mounts at the edges(preferably).

*Image after placing*



Routing and Filling:

Connect all the wires as done in schematic , use vias if direct not possible .



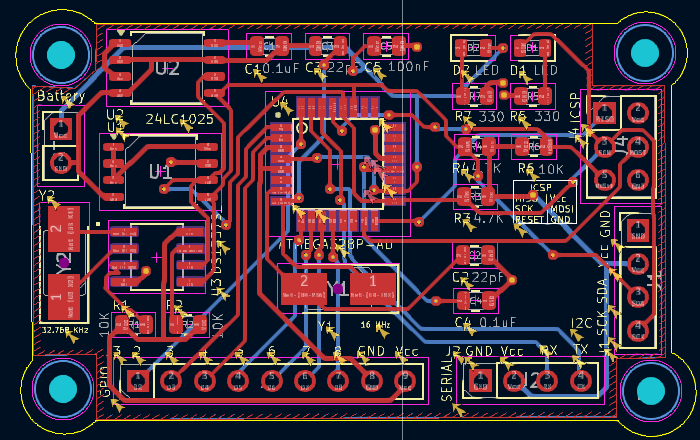
Do copper fils, first trace outline then fill all zone A red circuit board with many different colored lines

AI-generated content may be incorrect.

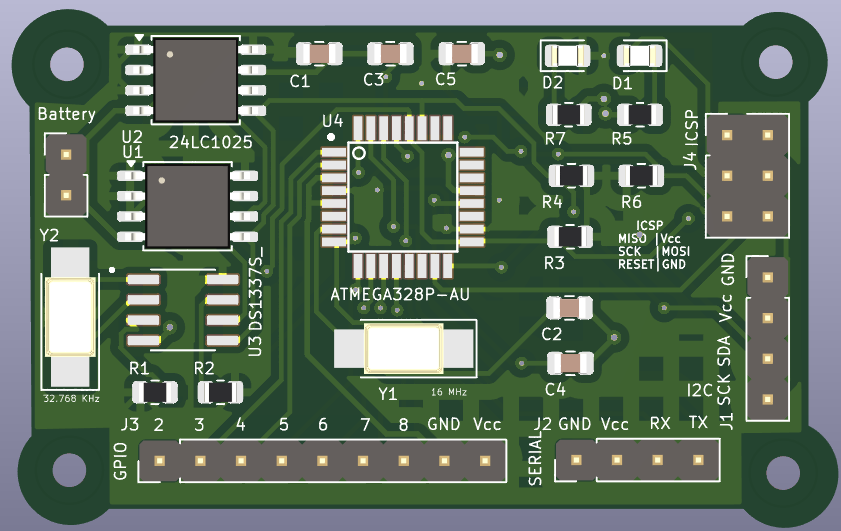
Run DRC, for design errors…

Write for silkscreen and add values and names to components

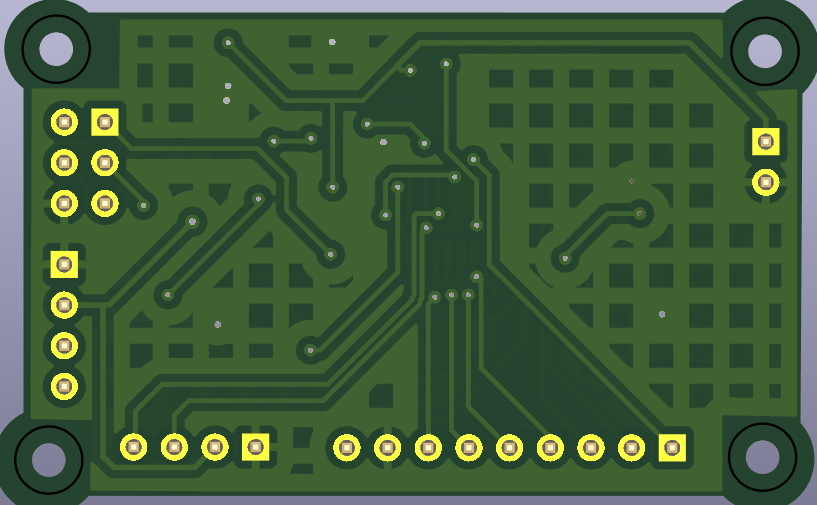
Do edge cuts also and re-arrange components for a smaller footprint(cost decreases)



FRONT:



Back:



FOR 4LAYER PCB

A computer screen shot of a circuit board

AI-generated content may be incorrect.

A circuit board with many wires and dots

AI-generated content may be incorrect.

A green circuit board with many small chips

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

To make changes to multiple branches:

Use git attributes.

git checkout main

vim .gitattributes

MCU Datalogger.kicad\_pcb merge=pcb //only this merges and updates in all branches

Git status

Git commit -am ”GA FILE”

Git add .gitattributes

Git merge main // merge changes only for mentioned file.