M-Track—— All-in-one outdoor sports computer

EECS 473 Fall 2022

Zhewei Ye, Zheyuan Wu, Yuantao Bai, Yifan Yang, Yidong Zhen, Yuxin Wang

I. Introduction and overview of project

Overview of Goal

We propose an all-in-one outdoor sports computer, the M-track, that offers map navigation and sports recording functions for outdoor sports lovers.

In our view, outdoor sports have two different needs: A) high-speed and short-term sports require high-precision exercise recording. B) low-speed and long-term sports require long-lasting navigation functions. A typical example of this is cycling: users want graphical navigation to confirm their position and the direction of the destination.

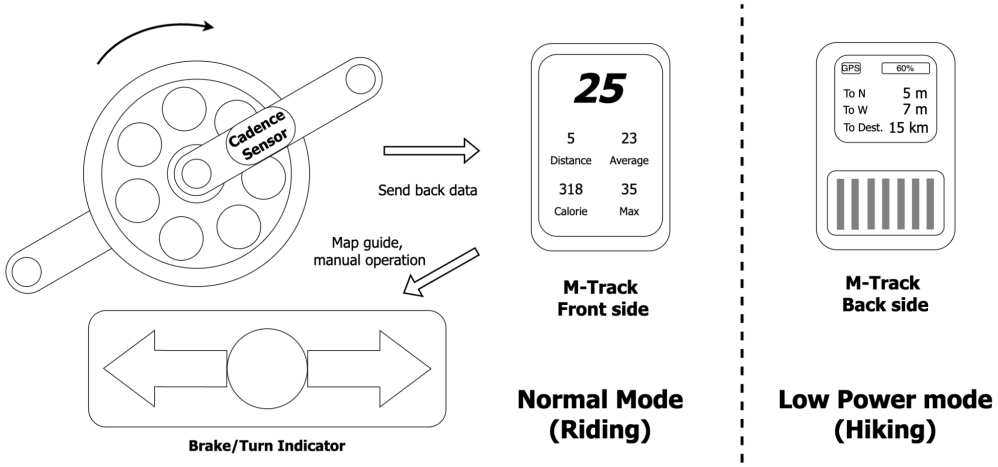


Fig 1. System Concept

Therefore, we separate outdoor sports into two typical scenarios and provide different modes. For sports like cycling, we provide a normal mode, which uses an LCD touchscreen for clear navigation and sports data display. We provide some customized components to provide more types of data or functionality. Here we include a cadence sensor and a turn/brake indicator provided for cyclists, which can enhance the sports experience. For sports like hiking, we will change to a low-power mode. These users can use a secondary E-ink mounted on the back to get succinct information. A solar cell is integrated to extend the M-Track's battery life.

There are some similar products on the market. For example, a Garmin 1040 bike computer costs $400 (Garmin). A device for hikers with long battery life costs another $400. Therefore, a low-cost alternative covering different sports has its market.

The M-Track can meet the needs of sports like cycling and hiking. It has the advantages of a lightweight (about 150g), acceptable price(lower than $100), long operation time (16-24 hours in normal mode and 30 hours to infinite in low-power mode), extendable comprehensive functions, and easy installation. After fully integrating the above functions, it surpasses the existing products on the market.

Expo Demonstration

The M-Track terminal enters its GUI after booting. It first enters the dial plate page, showing the sports information. The status bar above shows the time, number of satellites, and battery percentage. On the bottom are three buttons, which guide you to map settings, start workout toggle, and system settings.

Fig 2. M-Track's GUI

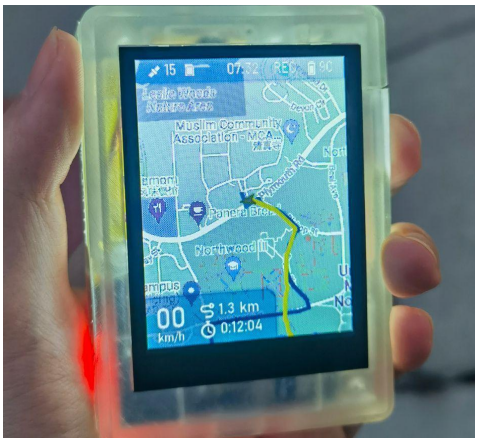


Fig 3. Navigation and Route Record

On the system settings page, there are 3 entries: system information, BLE, and WIFI configurations (unfinished). The system information provides raw data on the peripherals.

On the map settings page, there are 2 entries: live map and route selection. The live map can display the map based on the user's location. The route selection will list all available routes found in the TF card, and you can click to select or deselect. The selected route will be shown with Umich blue on the map.

If you start a workout, then your own route will be recorded and appear on the live map, marked with yellow. To facilitate user, user's route is above the navigation route and won't be blocked.

The M-Track has a normal mode and low power mode. This mode switch is provided when the user is in live map. The user can switch between two modes by double click on the encoder. Under low power mode, only the navigation route and the user's position arrow will appear on the screen. The user can refresh the E-ink by long press the encoder.

If you are cycling, the cadence meter could be installed on the crank to read the cadence. The data will be transmitted through BLE communication. The frequency of data broadcast is 10 seconds.

Fig 4. Navigation through LCD

Fig 5. Navigation through E-ink

Summary of Goals and Milestones

We managed to implement most of the functionalities mentioned in the proposal. Due to the tight schedule of teammates, we are not able to demonstrate: specific navigation instructions, cadence & brake module's PCB, Bluetooth integration with the terminal, and brake/turn light. In terms of milestones, we kept most designs and validation on track except for the brake light. This will be discussed in detail later in section III.

For milestone 1, we successfully made quick prototypes and were able to demonstrate LCD display, GPS positioning, and IMU sensing. Besides, we also finished the terminal's draft PCB schematics.

For milestone 2, we completed the migration of the open-source system framework on our quick prototype, finished the draft mechanical design, completed the terminal's PCB design, and sent it to the PCB fab. However, we are not able to demonstrate the map utility due to the delay in hardware design. Our cadence&brake module's hardware design is also delayed.

With that being said, we finally made a fully functioning terminal that met our expectations.

II. Description of project

System Concept and Feasibility

The goal of this project is to build an outdoor sports computer system that serves all kinds of outdoor sports. This system can provide navigation for users and can record users' sports data, work with some modules and provide specific data measurement. We provide a cadence sensor and brake/turn light module here to serve the biking situation. After a workout, the sports data will be stored on the TF card. Then the user can keep track of their sports performance.

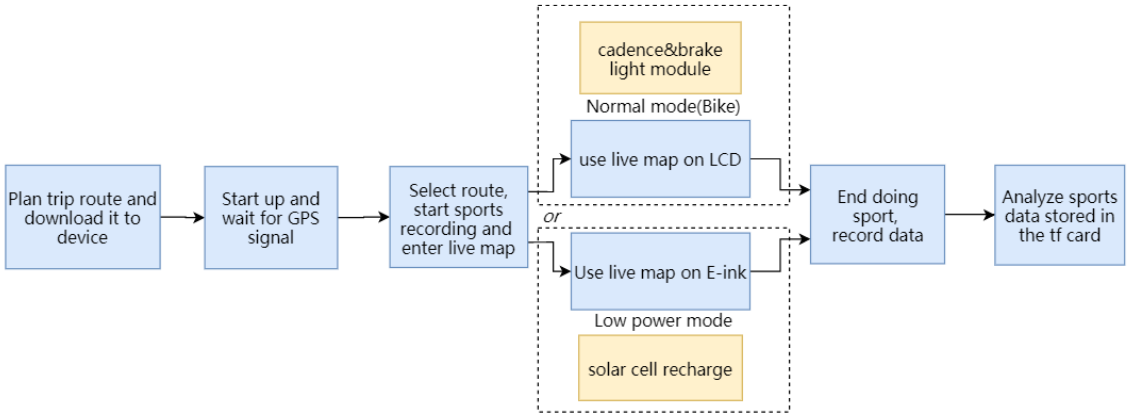


Fig 6. Flow Chart of the usage of the M-Track system

Constraints

The main constraints that we are concerned with were: the balance between weight and power consumption, the sensitivity of the sensors (GPS), the size of the device.

Firstly, we assume that the user always does their workout during daylight hours, and it will usually be sunny. The estimated battery life is 16-24h in normal mode and 40h to infinite in low-power mode.

Secondly, We choose a size of 61mm*88mm*27mm, and the total weight is around 150g. We assume that the users will be satisfied with it. Similarly, we assume that a 43mm*44mm cadence won't interfere with the performance of the cyclists.

Thirdly, we assume that users will use it in an open area without blocking the GNSS signal. Our main functions are based on positioning, so a suitable GNSS (Global Navigation Satellite System) module is important. We choose the ATGM336 module, which supports Beidou, GPS, GLONASS, and GALILEO. It can receive 17 satellites' signals which can reduce cold start-up time and improve accuracy.

System Architecture and Detailed Description

Our system contain 3 devices: The M-Track terminal, the cadence sensor, and the brake/turn light module. the cadence &brake/turn light module share the same PCB while using different peripherals.

Here we introduce our system architecture from the hardware and software of the M-Track terminal to the hardware and software of the cadence& brake/turn light modules.

Hardware design of M-Track terminal

1) Part selection and data flow chart

It is composed of a microcontroller, two screens(LCD and E-ink), a GNSS module, an IMU sensor, a TF card slot, power management which supports the solar cell and type-c, a battery and an encoder.

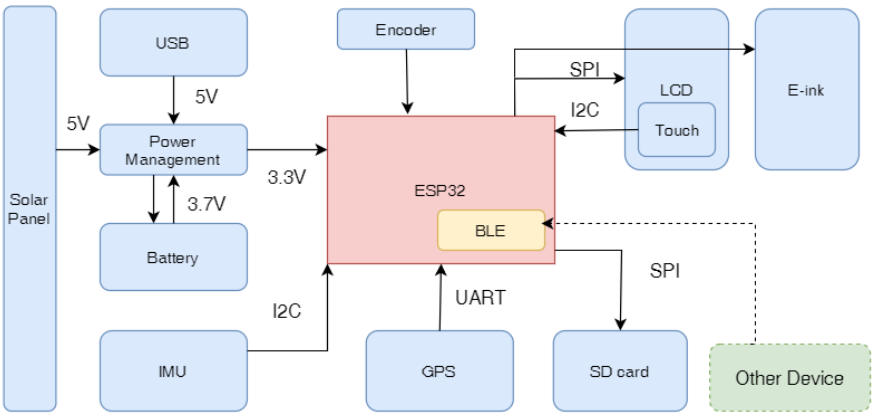


Fig 7. Flow chart of the M-Track

To accelerate the development progress, we select ESP32-PICO-D4, which integrates a crystal oscillator, and flash in one package. It also supports the Arduino platform, so we have abundant libraries to use.

We use a 2.8inch 320*320 LCD screen which also supports capacitive touch. This screen is large enough to display information clearly, and the PPI is not too high, which won't give MCU too much burden.

We integrate the bq24074 power management chip provided by TI. It supports power input from solar cells and type-c. It can monitor battery status and delivers voltage output without harming the battery.

2) Mechanical design

To make our design more intuitive, we put the LCD on the top side, and E-ink and solar cell on the bottom. When users are in normal mode, they would face the LCD. In low-power mode, users would get information from the E-ink, and the solar cell wouldn't be blocked.

Then we walk inside. To make the position of the encoder friendly, while using the least area of PCB, we separate the PCB into two parts. The main board would have most utilities. The encoder and antenna are on the sub-PCB, which is placed on the top right in the inner space.

This primitive mechanical design confirms the size of the PCB, enabling PCB design.

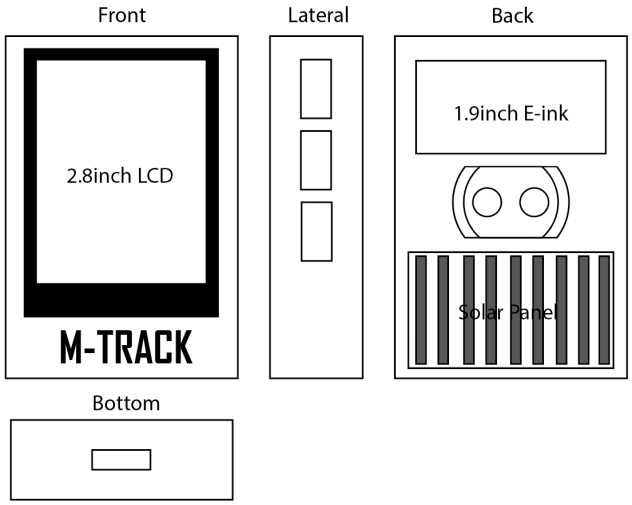


Fig 8. Mechanical design (outer view)

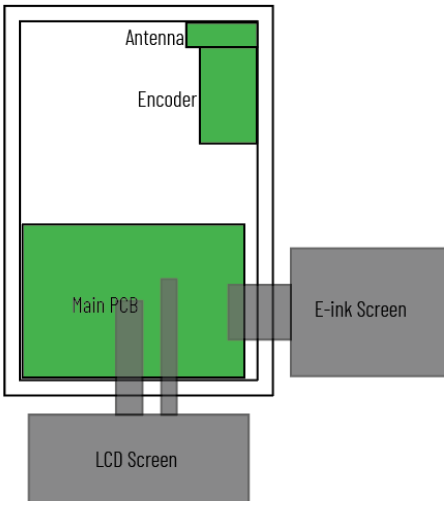


Fig 9. Mechanical design (inner view)

3) PCB design

We believe the key for students like us to do PCB right is to be good copycats. We learn and copy chips' dev-board schematic and PCB design. Then we combine them together and add some features, for example, the p-mos switching used to shut down some peripherals to achieve low-power feature.

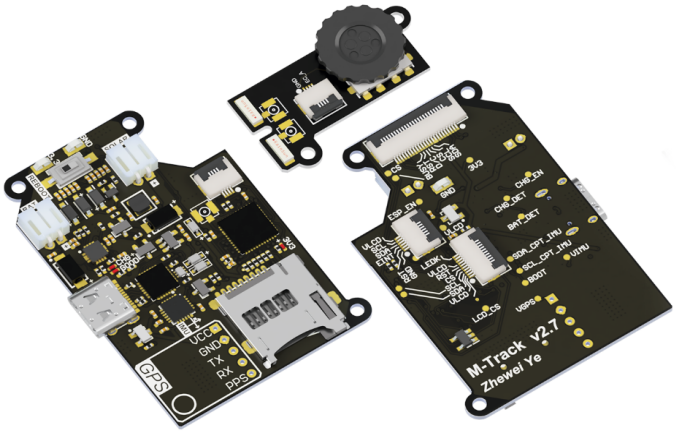


Fig 10. The M-Track's PCB

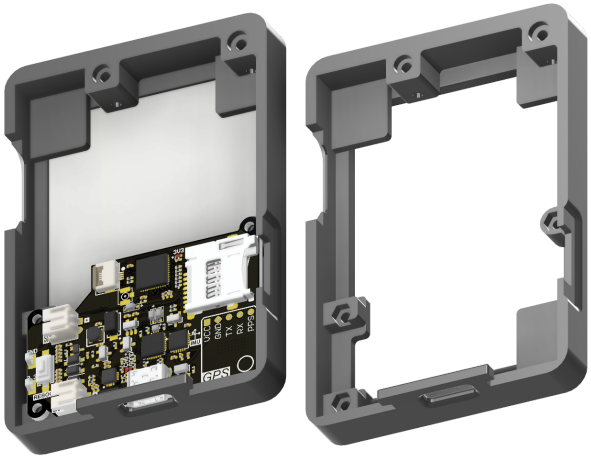


Fig 11. Top Case (inner view)

To make it easy to debug, we add sufficient silkscreen and test points on the bottom. The design also concerns power integrity, signal integrity, and mechanical friendliness.

4) 3D printing case

The 3D printing cases have a top half and a bottom half. Because the precision of printing is not stable, we avoid modeling the screw thread directly and use m2 nylon screws and nuts to fasten.

The main PCB board and LCD are bound on the top case. We also leave holes for the Type-C, encoder, and TF card. During the iteration, we reduce the side wall's thickness from 4mm to 2.5mm to improve the operation feeling of the encoder and make the insertion of TF card and Type-C male head easier.

The sub-PCB, the Battery, E-ink, and the solar cell are located on the bottom case. We designed a slot to mount the delicate E-ink. The antenna is on the top right to get a good signal. The sub-PCB is connected to the main PCB with an FPC cable and coaxial cables. We here select a 2800mAh battery for long battery life.

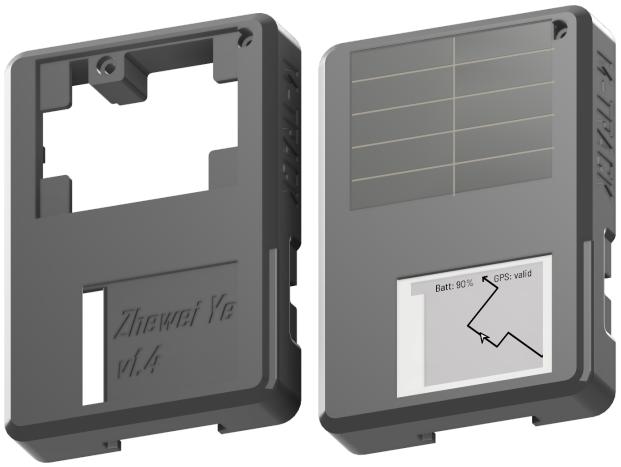


Fig 12. Bottom Case (inner view)

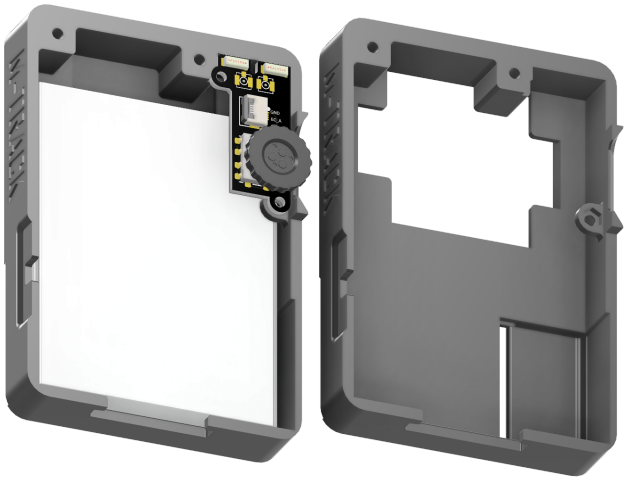


Fig 13. Bottom Case (outer view)

5) Assembly

Since we put a lot of effort into the entire design flow and made many revisions, the assembly is quite satisfactory.

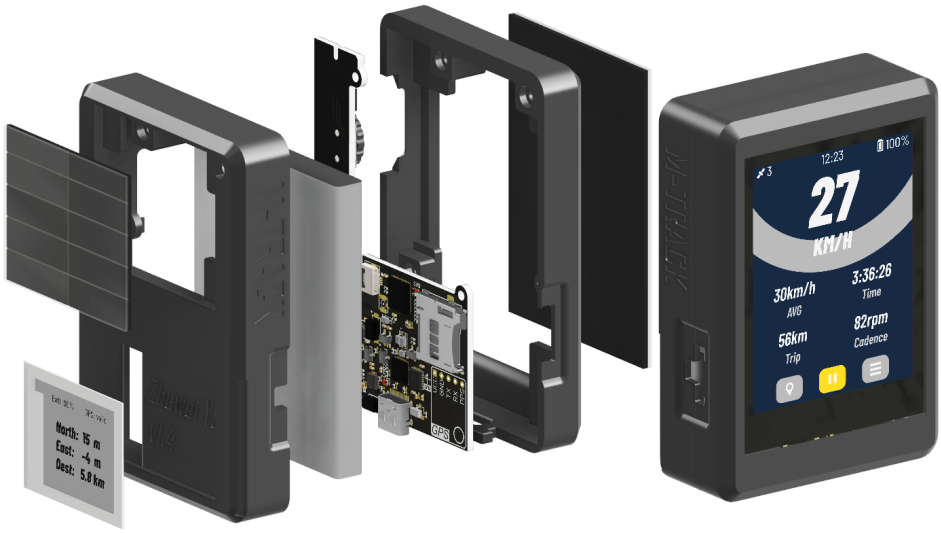


Fig 14. Assembly of the M-Track

Software of The M-Track terminal:

We migrate the X-Track's system framework here and bring up features based on this gorgeous open-source project.(see section VIII) This software uses the notion of abstraction. The software is divided into 3 layers: HAL(hardware abstraction layer) layer, Framework layer, and APP layer.

The HAL layer encapsulates the API of the drivers and provides basic service for upper software.

The system framework layer contains LVGL(Little Versatile Graphic Library) which provides graphic utility and file system, The page management which uses a stack to store the page history, and a data buffer which collects data and provides them to subscribers in the system. These modules provide service for the apps we would like to use.

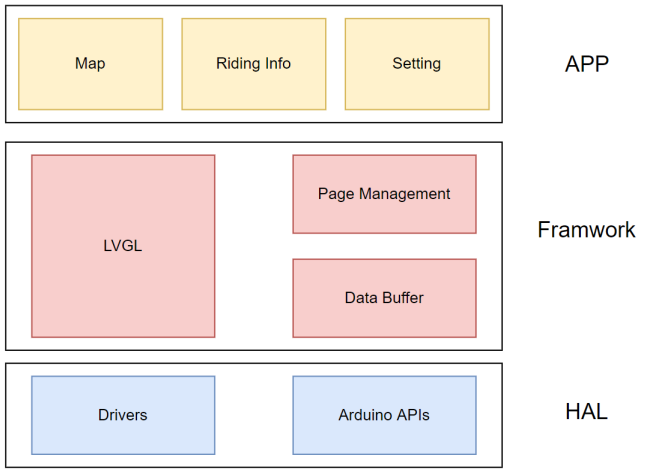


Fig 15. Software Structure

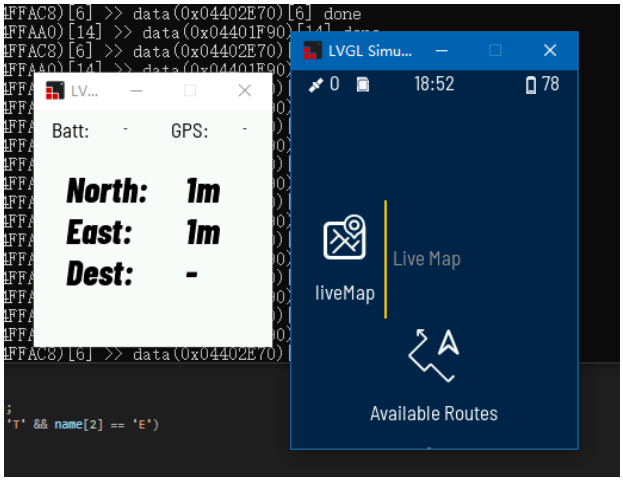


Fig 16. Software Simulator

The apps in the APP layer are some pages. They are designed to satisfy our request: information display and navigation. The specific pages mentioned above in Expo demonstrate, section I.

The M-Track software can also be developed using a simulator. By providing a pseudo-HAL layer, the system can work on a PC. This can greatly improve development efficiency.

1) Mode Switch:

The key of the mode switch between Normal mode and Low-power mode is the switching between LCD and E-ink. This contains two parts, the LVGL configuration (to support drawing content on two screens), and making the screens' drivers compatible with each other.

For the LVGL part: the LVGL supports different sets of the display driver, we can use global variables to record the screen object which corresponds to a screen driver. Then, the user can add and edit objects bound with the screen object, and the LVGL will update the content on that screen using a proper driver.

For the driver of two screens: the screens are sharing the same VSPI bus, and the libraries are using the same SPI object. The SPI lib uses a semaphore to lock the hardware resource if it is using SPI. Before SPI begins the transaction, it will check whether the SPI is being used, and lock the system by trying to take this semaphore in a while loop. The original E-ink library doesn't end its SPI transmission after initialization, when the LCD driver tries to start the SPI transaction, the system will die. So we fix the E-ink library by ending the transaction every time E-ink sends a package.

The reinitialization of E-ink and LCD logic is trivial, so we skip it.

The last step is to embed the E-ink output into the original system. We decide to embed it in the map refresh service. There is a timer that will check and update the map at a certain interval. We use the LVGL and encoder library to detect a certain click pattern(double click) and during updating of the map, it will switch to E-ink and update content on the E-ink, then come back to LCD. So that the map service is nearly intact, and can provide its service to LCD or E-ink.

2) Navigation Route:

The navigation route feature can be separated into 3 parts: enumerating files in TF card, selecting and deselecting a certain file using GUI, Showing the route on Map and on E-ink.

We create a Route class and create some static variables in this class to record the route selection information. We enumerate the route files in a certain directory after booting and store the information in the Route class.

We add a page for users to view and select the routes. The pages have this Route object and can read the variables to get the route selection. After clicking on a certain route widget, the widget will alter its text to “selected”, showing the selection, and clicking again will cancel the selection.

Then we introduce how to show the route on the live map. When loading the map page, it will check the selection choice, load and parse the file, and store the route information using a vector. Then, it will create an LVGL line object and convert the latitude and longitude data into the coordinate used by the Map Tile system, then draw the line on the map. The map will update the route based on the move of the user. Update the route displayed on the E-ink is not exactly the same. When we are using the live map in normal mode, the environment is depicted using pictures stored in the TF card. The Map Tile system will choose the proper picture by latitude, longitude, and zoom level. The points will be converted relative to the origin of this picture. Then, the user's movement will be regarded as moving of the map picture (the user is always in the center of the screen). But the E-ink only has 2-bit color. and is hard to show map information. So we decided only to depict route information and user course direction on E-ink. When updating, we rotate the arrow located in the middle of the E-ink, calculate the relative offset between the user and route, and use this offset to depict the route on the E-ink.

Hardware design of Cadence & Brake modules

1) Part selection and data flow chart

The hardware design of cadence and brake light are the same, and they use part of the M-Track terminal’s design. The cadence sensor only uses the IMU, while the brake light uses both IMU and LED Matrix as output.

We use BLE to support a communication link among the M-track terminal, the cadence sensor, and the Brake module. The M-track terminal acts as an input and output device, fetching data from the cadence meter, and at the same time giving commands to the brake light.

2) Cadence sensor

The cadence sensor is controlled by an ESP32, and data is measured by an MPU6050 accelerometer.

MPU6050 can serve as a 3-axis gyroscope or 3-axis accelerometer. We use it to measure the angle of the wheel crank. We use inverse trigonometric functions to calculate the angle. With the angle being calculated, we can count the number of rounds it has turned and calculate cadence.

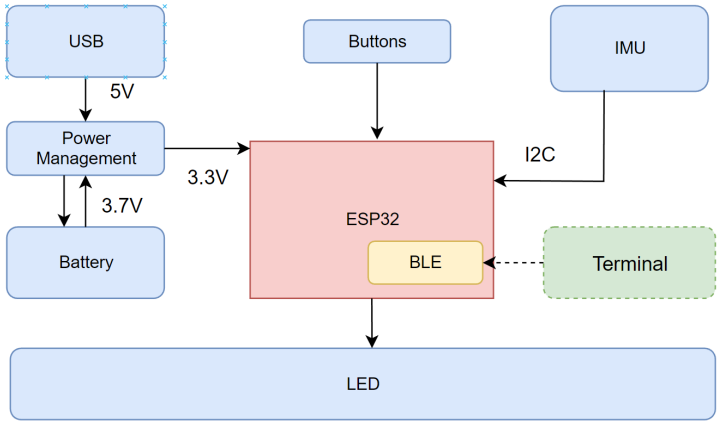


Fig 17. Flow chart of Cadence & Brake

3) PCB design

The cadence sensor and brake light share the same PCB designs to reduce costs. The PCB consists of power management, USB data management, IMU, and MCU. These designs refer to the M-track board. However, due to the soldering problem, the final board could not work properly, so we use the solder board instead in Expo. The possible reasons will be discussed in session IV.

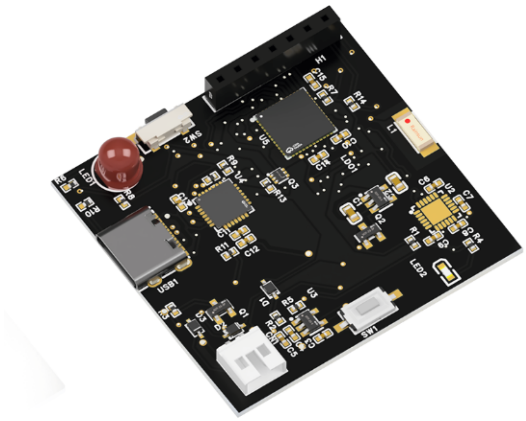


Fig 18. The PCB of Cadence &Brake

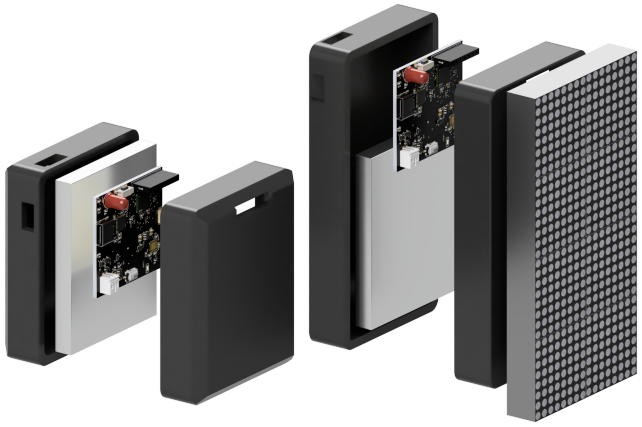


Fig 19. Assembly of Cadence &Brake

4) 3D printing case

Two 3D printing cases are designed to hold the cadence&brake light respectively. We use buckle design instead of assembly by screw threads.

The cadence PCB is designed to be placed on top of the battery. The button case has a 1mm buckle edge, which enables the top case to be buckled on it. We leave holes for the Type-C port.

The case of the LED light is longer to fit the size of the LED matrix. The battery could then be installed beside the chip as there is more space. A 20mm x 3mm port is left for the LED matrix IO port.

5) Assembly

In the real case, we use the solder board instead of the PCB. After many revisions are made, the assembly of both cases meets expectations.

Software design of Cadence & Brake modules

The cadence sensor needs to communicate with the main terminal to send the cadence data to the terminal, while the brake light should also have a communication link with the terminal to get command of which light to turn on.

We set the main terminal as the server and set both the cadence sensor and the brake light as clients. The server can advertise its existence and carries the data that the client can read. The clients can scan the nearby devices, and when they find the server by UUID, it establishes a connection.

As a server, the main terminal holds its own service and characteristic, and the characteristic is shared with its clients. The information consists of the cadence data and the left or right command. The cadence client can access this remote characteristic, and edit the information. The M-track terminal could then extract and read the cadence data. The brake light client can get access to that characteristic information and control the LED based on the command.

Low-power testing

power consumption test

1) Power consumption of different mode with GPS on

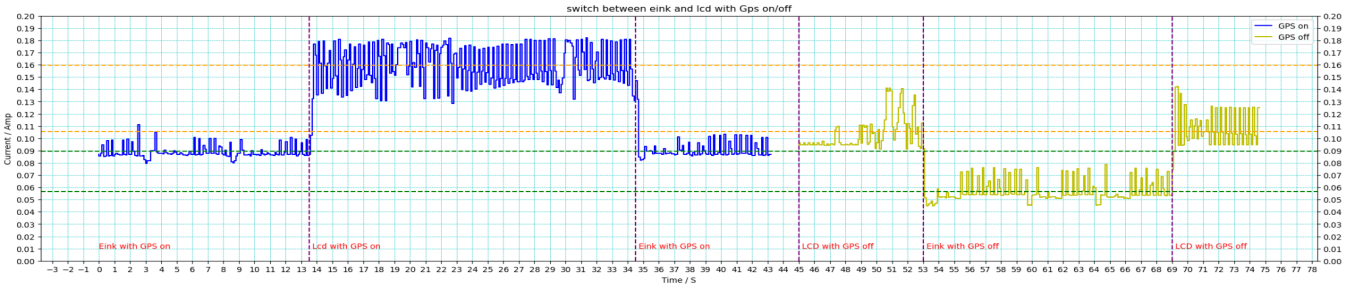


Fig 20. demonstrates the change in current of the M-Track when switching between display modes with GPS on/off. With GPS working, the average working current of the M-Track with Lcd is 0.159A (orange line), and the average current when using E-ink is 0.0895A (green line). With GPS off, the average current of the M-Track when using Lcd is 0.1057A (orange line), and the average current when using E-ink is 0.056A (green line).

2) Statistic analysis of power consumption

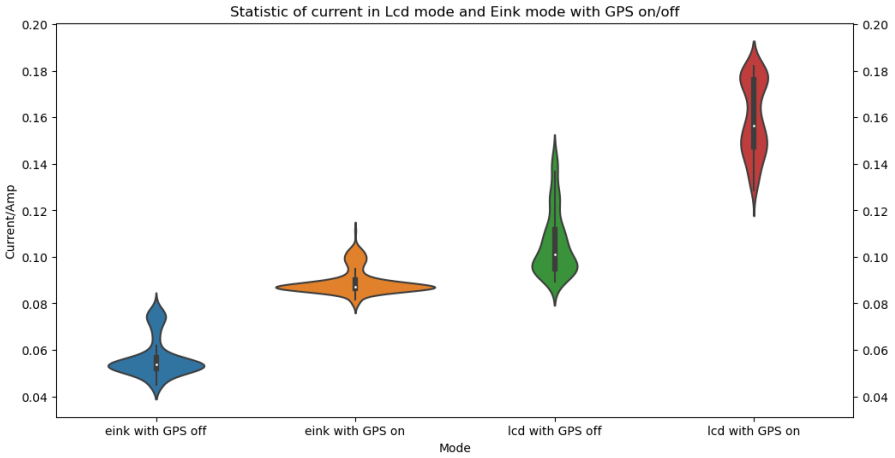


Fig 21. the statistics of the M-Track's current when using E-ink/Lcd and GPS on/off. The difference between the average power consumption of different modes is greater than twice the std deviation. Therefore, according to the null hypothesis, reducing the GPS operating frequency and switching to E-ink play a positive role in reducing system power consumption.

In addition, as shown in Figure, the operating current presents a bimodal distribution when using E-ink for display. This is because E-ink only consumes power when refreshing, and does not consume power to maintain the display.

Charging test

1) Charging standby-mode the M-Track with solar panel

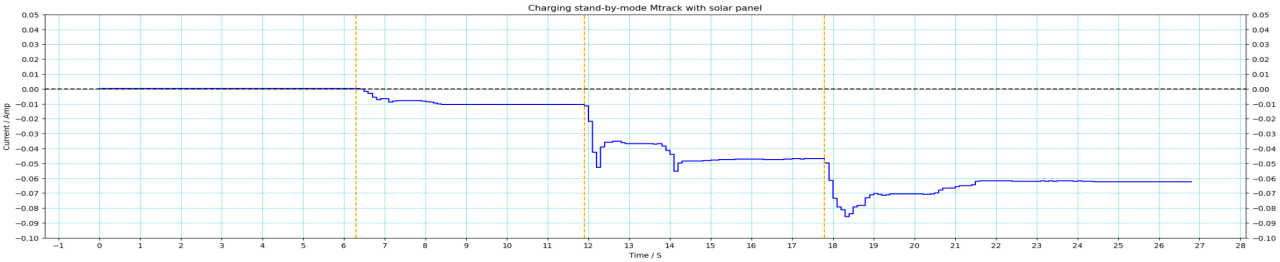


Fig 22. Itshows the result of charging standby-mode the M-Track using the solar panel. At time 6, the solar panel is illuminated with an incandescent lamp. At time points 12 and 18, the light intensity was increased stepwise to simulate different outdoor lighting environments. It shows that the solar panel can provide a charging current of about 60mA under strong light, and 47mA charging current under moderate light.

2) Charging the M-Track with solar panel in different modes

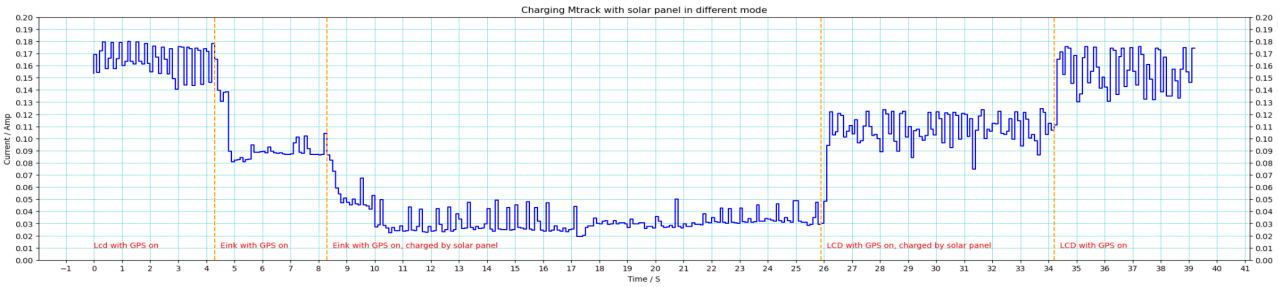


Fig 23. it demonstrates the battery supply current when a solar panel is illuminated with a strong light in different modes. In time period 0-4, the M-Track is working in Lcd mode with GPS on and is not powered by solar panel; in time period 4-8, the M-Track is working in Lcd mode with GPS on and is powered by solar panel; in time period 8-26, the M-Track is working in E-ink mode with GPS on and is powered by solar panel; in time period 26-34, the M-Track is working in E-ink mode with GPS on and is not powered by solar panel; in time period 34-39, the M-Track is working in Lcd mode with gps on and is not powered by solar panel;

Ablation experiment

Solar Panel	low power (E-ink)	GPS off	Battery Current/mA
✓	✓	✓	-6.4(estimated)
✓	✓		30.8
✓		✓	59.23 (estimated)
✓			108.5
	✓	✓	56.6
	✓		89.3
		✓	105.7
			159.5

This design uses a 2800 mAh battery. In normal mode, the average working current is 160 mA, which is 0.06C. The estimated available power is 2600 mAh, and the working time is about 16 hours; In low power mode with Low GPS working frequency, the average current is 60 mA, and it can work for 43 hours without relying on solar panels for power supply; in the case of sufficient sunlight, the power supply current of solar panels is more than 60 mA, the M-Track can theoretically achieve infinite Battery life.

III. Milestones, schedule, and budget

Milestone 1

For milestone 1, our deliverables were:

1. Research for similar designs
2. Get familiar with its development platform and do some demos.
3. Design the software framework
4. Demos of LCD display module and E-ink display module

5. Finish M-Track's PCB draft schematic

We did most of the research and design, successfully made a quick prototype and were able to demonstrate LCD display, GPS positioning, and IMU sensing. Besides, we also finished the terminal's draft PCB schematics(v1.0) and the draft mechanical design. We were stuck by the part selection of power management. At the milestone meeting, Prof. Mark and Matt recommended the MCP73871 dev board, and based on this, we found the bq24074, which is more compact and fit our project better.

We didn't manage to design the API for the map component. The reasons are: our brainstorming took too much time. Some of the members stuck by the labs, HW1, and several midterms.

We spent $186.17 on evaluation boards by far. The budget spent stayed within our expectations and was considered healthy at the time.

Milestone 2

For milestone 2, our deliverables were:

1. Migrate the open source framework.
2. Designed and modified the GUI, deploy it on the hardware, and make sure that it can do some basic jobs.
3. Get familiar with the system framework's modules, write documentation, and brainstorm how to bring up the features.
4. Finished the terminal's PCB design, communicate with the Fab, matching the components that were used in PCB assembly.
5. Complete the terminal's mechanical data calculation, and build the terminal's case 3D model.
6. Complete the demos of the LCD display module and E-ink display module.
7. Integrated the peripherals demo, making them work together.

By milestone 2, we completed the migration of the open-source system framework on our quick prototype and are able to demonstrate some pages. We finished the mechanical design and began to do case 3d modeling. We also finished the terminal's PCB design(v2.7) and sent it to PCB fab.

We were not able to demonstrate the map utility due to the delay in hardware design. Our cadence &brake module's hardware design is also delayed. The reasons are: some of the members were stuck by their workload and interviews, so there are some delays in cooperation.

We spent $869.32, including $294.19 for PCB fabrication. To make the PCB more compact, we chose 0402 resistors and capacitors, and chips with QFN package. We thus used PCB assembly service to substitute hand soldering as it can largely lessen our burden.

Summary

In the end, our team overspent the budget by $126.13. The reasons are attached below:

1, Mistake in mechanical design, Zhewei wrongly estimated the 2.13in E-ink's FPC cable's length, so he have to buy the extend cable and use a 1.54in counterpart to replace it. (Caused around $80.)

2, PCB assembly cost more than expected. We expected our boards would use $360 as the total, but the actual spending is $295(M-Track terminal) + $186(Cadence &brake) = $481.

3, 3D printing prototyping. Zhewei built several prototypes (v1.2 and v1.3) before the final version(v1.4), and the printing of the cases cost $40+$40 (two prototypes+final), which is not included in the proposal.

IV. Lessons learned

Lessons from Success

Utilize open-source resources: before deciding to develop some features, search for GitHub or other platforms. We referenced X-Track, Peak, and Peak-T2 during part selection and software feature development. We learned a lot from their system design, interface design, and code style.

Follow the design process: we followed the design—prototype—iteration process, by producing prototypes, and we gradually enhanced the model. The prototype is really important for it makes some design flaws obvious. For example, the first 3D printing prototype's side wall is too thick for the encoder to press. So in the final version, we reduced the thickness, and the operation feeling is improved.
Using GitHub version control and branch: I have several branches which have their own usage: main branch which is the whole code, demo branch which is demos of peripherals on the quick prototype, demo_for_MTrack which is demos working on M-Track hardware, and preDemo_Bluetooth which is used to test the integration of BLE with M-Track software.
The M-Track PCB with full functionality: there are many iterations of PCB before the final version (v2.7). I double-checked the schematic while asking friends who specialized in PCB design and 473 staff to help me review the PCB. The PCB went out to have nearly full functionality, except I mistakenly chose the wrong resistor value. Replacing it fixes the functionality.

Software features bring up: the software feature bring-up contains 70% of the understanding of system framework and design, and 30% of code development and debugging. Most of the time spent is used to understand the logic of the software design of the open-source project. I practice and validate some ideas on the demos branch, and integrated them into the main branch.

Don't panic and be impetuous, look into the source code: I stuck with the LCD&E-ink switching feature for a long time. The LVGL doesn't provide an official example of display switching, and the LCD library and E-ink library seem to have conflicts: switching may make the system dead. I used LOG to gradually locate the codes that caused errors and analyze them. Finally, I found that one of the display libraries locks the SPI without unlocking it. Modifying the lib fixed my problem(mentioned in section II).

Lessons from Failures

Small issue in PCB: We failed to make the Cadence & Brake module PCB work. Its schematic is a subset of the M-track. As the main PCB works with full functionality, it may be the problem of soldering the QFN48 chip. The chip has pins underneath, which makes it quite difficult to solder, test and debug. We need to ask for professional service from PCB printing to solder this chip instead of soldering on our own next time. However, if we have to solder such a chip, we need to solder this first on board instead of leaving it as the last item to solder.

Technical Material Learned

Zhewei Ye had prior experience from his undergraduate Formula Student Electric team and some graduate courses, so he has experience with hardware design & debugging, and algorithms. This time, he was responsible for everything related to the M-Track terminal. He learned about 3D modeling skill, skills to migrate the software system from an open-source project to his own project and extend an original project, concepts about the software system layer design, and tricks to incorporate two libraries using the same SPI bus by looking into the source code. All in all, it was a valuable experience

for him to be exposed to full-stack embedded project development from PCB design to the software system and map navigation.

Zheyuan Wu used to have experience in PCB design for the power converter. In this project, he designed how to add a microprocessor into the PCB and how to design power management between the type-C port and battery. Also, at first, Zheyuan Wu designed devices as both server and client, which make the code quite complex and unreasonable. After he went through the material of the BLE, he learned the structure of BLE and figured out that one server and multiple clients sharing the same information will be a better solution.

Yifan Yang learned how to make a BLE connection between two ESP32 and then how to do Max7219 8x8 LED Matrix scrolling text control using BLE. He also learned how to use fusion to build 3D models of cadence. When using a 3D printer to print, he encountered some problems such as not extruding, overheating, and stopping extruding mid-print, then he started to go over 3D printing materials to figure out how to change software settings to solve these problems. Finally, the issues have been addressed.

Yuantao Bai learned how to integrate independent hardware interface libraries. The graphical interface library LVGL used in the project is highly mature, and the E-ink driver library is focused on the interface program at the hardware level. Using the two at the same time will encounter the problem of functional crossover or even conflict. YuantaoBai refactors the E-ink interface program to adapt to the LVGL. This experience allowed YuantaoBai to learn how to extend new hardware in the existing software system.

Yidong Zhen learned how to design the GPS tracking module, including connecting the GPS positioning with the route navigation. Also, he learned how to design power management and make it work. He also learned the mapping documents protocol and design. From reading and changing the code according to his own needs, he consolidated his programming skills. Overall, through this project, Yidong has a deep understanding of the entire process of designing and manufacturing a PCB project.

Yuxin Wang learned how to design a complex PCB and what to be paid attention to in actual operation. Yuxin Wang also learned how to solder PCBs, especially how to solder small chips on complex PCBs. He learned how the accelerometer works, the working principle of the cadence meter, as well as the algorithm applied and installation methods. He learned how to use Autodesk Inventor software to create the second Case. Through the use of 3D printers, he learned how to do a successful print job.

V. Contributions of each member of team

Team member	**Contribution**	**Effort**
Zhewei Ye	Zhewei Ye was responsible for everything related to the M-Track terminal, from part selection, mechanical design, PCB design and debugging, case 3D modeling, software migration to features bring-up. He developed the mode switching and the navigation feature. He wrote technical documentation that contains tutorials, resources, and design detail. He used GitHub to manage source code and other things. He used the PC simulator to debug the software. He worked out the dual-screen simulation on PC. He helped to review the cadence&Brake module's PCB. He also did the 3D model	50%

	rendering, and the pictures are used to present the work. He designed the Expo poster and wrote the related part of the report.	
Zheyuan Wu	Zheyuan Wu took charge of the cadence sensor build-up. He designed an algorithm to read the turning angle of the wheel with the accelerometer, thus figuring out the cadence of the bike, and then uploaded corresponding data onto the main device (sports computer). He designed PCB for the whole cadence system to optimize the size of the device to make it fit on the pedal. He also built the BLE communication link among the sports computer, cadence sensor, and brake light. He wrote the cadence & brake module part in the report.	15%
Yuantao Bai	Yuantao is mainly responsible for the E-ink module in the project. He optimized the interface program of the E-ink module, realized the display function of the E-ink module, and adapted it for LVGL. Yuantao has completed the switching process from LCD to E-ink, and deployed Free-RTOS in the product for multi-task management and screen switching. Yuantao also completed the reading and writing of the SD card and assisted other team members in debugging and deploying the cadence module.	15%
Yifan Yang	Yifan Yang is responsible for the turn/brake indicator module which displays turn signals under the control of BLE signals. He was in charge of the BLE server setup. Besides, he built the 3D model of cadence and used 3D printing to print the crust of cadence. Yifan was also in charge of measuring the current consumption of different modules of the Bike computer and assisted teammates in testing and debugging.	10%
Yidong Zhen	Yidong is responsible for the selection of LCD and E-ink screens as well as solar panel. He designed the GPS tracking module and connected the solar panel to the system power. Besides, he helped achieve the map API design and map navigation module for the system to generate the navigating route. He also assisted other team members in bike cadence installation and M-track assembly.	5%
Yuxin Wang	Yuxin Wang is in charge of GPS chip selection and power consumption tests. He participated in the cadence &brake module PCB design. He completed the above-mentioned PCBs soldering work. He was responsible for the second set of 3D case designing and printing. He and other members assembled the cadence &brake light to conduct on-site Bluetooth connection tests and corrections.	5%

VI. Cost of Manufacture

The fabrication cost of our PCB (54mm * 51mm, 4 layers, including assembly) from jlcpcb.com is $17.42/pc, given the population is 1000 pieces. For the PCB components, the prices for 1000 pieces are shown in Table 1 below. The estimated cost would be $36.655/pc.

Footprint	Matched Part Detail	Qty	Total Cost
CAP_0402	CL05B104KO5NNNCC1525 16V 100nF X7R ±10% 0402 Multi...	7010	$4.21
CAP_0402	0402B222K500NTC1531 50V 2.2nF X7R ±10% 0402 Multi...	1005	$0.70
QFN-28	CP2102-GMRC6568 Transceiver 1Mbps USB 1/1 QFN-...	1000	$1,419.00
SOD-323	B5819W SLC8598 40V 600mV@1A 1A SOD-123 Schot...	3001	$50.12
FPC0.5-WT-10P	C11085C11085 Slide Lock 10 Contact,Top Surf...	1000	$51.00
RES_0402	0402WGF1001TCEC11702 62.5mW Thick Film Resistors 50...	2005	$0.60
SOT23-3	AO3401AC15127 30V 4A 44mΩ@10V,4.3A 1.4W 1.3V...	3002	$153.70
CAP_0402	CL05B103KB5NNNCC15195 50V 10nF X7R ±10% 0402 Multil...	1005	$0.60
CAP_0402	CL05A106MQ5NUNCC15525 6.3V 10uF X5R ±20% 0402 Multi...	9005	$28.82
RES_0402	0402WGF0000TCEC17168 62.5mW Thick Film Resistors 50...	4010	$1.20
SOT23-3	AO3400AC20917 30V 5.7A 26.5mΩ@10V,5.7A 1.4W ...	2001	$129.66
CAP_0402	CL05A475MP5NRNCC23733 10V 4.7uF X5R ±20% 0402 Multi...	7005	$23.82
QFN-24	MPU-6050C24112 QFN-24_4x4x05P Attitude Senso...	1000	$8,643.00
RES_0402	0402WGF100JTCEC25077 62.5mW Thick Film Resistors 50...	1005	$0.30
RES_0402	0402WGF2000TCEC25087 62.5mW Thick Film Resistors 50...	1005	$0.30
RES_0402	0402WGF1003TCEC25741 62.5mW Thick Film Resistors 50...	4010	$1.20
RES_0402	0402WGF1002TCEC25744 62.5mW Thick Film Resistors 50...	18008	$5.40
RES_0402	0402WGF1202TCEC25752 62.5mW Thick Film Resistors 50...	2005	$0.60
RES_0402	0402WGF2002TCEC25765 62.5mW Thick Film Resistors 50...	2005	$0.60
RES_0402	0402WGF5101TCEC25905 62.5mW Thick Film Resistors 50...	2005	$0.60
CAP_0402	CL05A105KA5NQNCC52923 25V 1uF X5R ±10% 0402 Multila...	6005	$10.81
QFN50P300X300X1...	BQ24074RGTRC54313 QFN-16_3x3x05P Battery Manage...	1000	$1,305.00
LED_0402_RED	16-213/SDRC/S530-A3/TR8C71911 Red 0402 Light Emitting Diode...	1001	$29.43

SOD3716X135N	MBR0530C77336 30V 550mV@500mA 500mA SOD-123 ...	3001	$54.02
SOT23-5	ME6215C33M5GC84119 350mA 320mV@(200mA) Fixed 3.3V...	1001	$56.06
SMA	SK34A-LTPC85269 40V 3A 500mV @ 3A SMA Schottky...	2001	$92.85
ANT-Rainsun-AN5... .	AN2051-245C99727 2051 Antennas ROHS	2000	$585.00
LED_0402_RED	LTST-C281KGKTC160479 20mA 574nm Colorless transpare...	2001	$78.04
FPC0.5-WT-4P	C191195C191195 SMD,P=0.5mm FFC/FPC Connector...	2001	$129.66
QFN-48_7X7	ESP32-PICO-D4C193707 QFN-48_7x7x05P RF Transceiver...	1000	$2,626.50
SOT563-6	EMX1T2RC253326 100nA 50V 150mW 150mA 120@1mA,...	1001	$37.14
SW SMD-3*6*2.5	TSA363G25-250BC354943 No NO Gull Wing 50mA 6.05mm 10...	1001	$30.33
SOT95P285X140-5... ...	74LVC1G04W5-7C460517 SOT-25 Inverters ROHS	1001	$65.77
SOT65P210X110-3... ...	SI1308EDL-T1-GE3C469327 30V 1.4A 400mW 132mΩ@10V,1.4A ...	1000	$337.50
FPC0.5-WT-6P	FPC05006-09200C479749 Clamshell -25℃~+85℃ 6 Bottom C...	1000	$71.30
FPC0.5-WT-24P	FPC05024-17205C718080 -45℃~+85℃ Clamshell 24 Double-...	1000	$174.30
Type-C-TOP	TYPE-611-T3C2689843 3A 1 Surface Mount 16 Female -...	1000	$56.10
TF-SOCKET	TF-012DC2874207 2mm Deck MicroSD card (TF card...	1000	$78.20
SIQ-02FVS3	SIQ-02FVS3C2925423 SMD Pre-ordered Products ROHS	1000	$1,447.50
COAXIAL/SMD/CON... ...	IPEX-4C2987683 SMD RF Connectors / Coaxial C...	3001	$229.58

VII. Parts and Budget

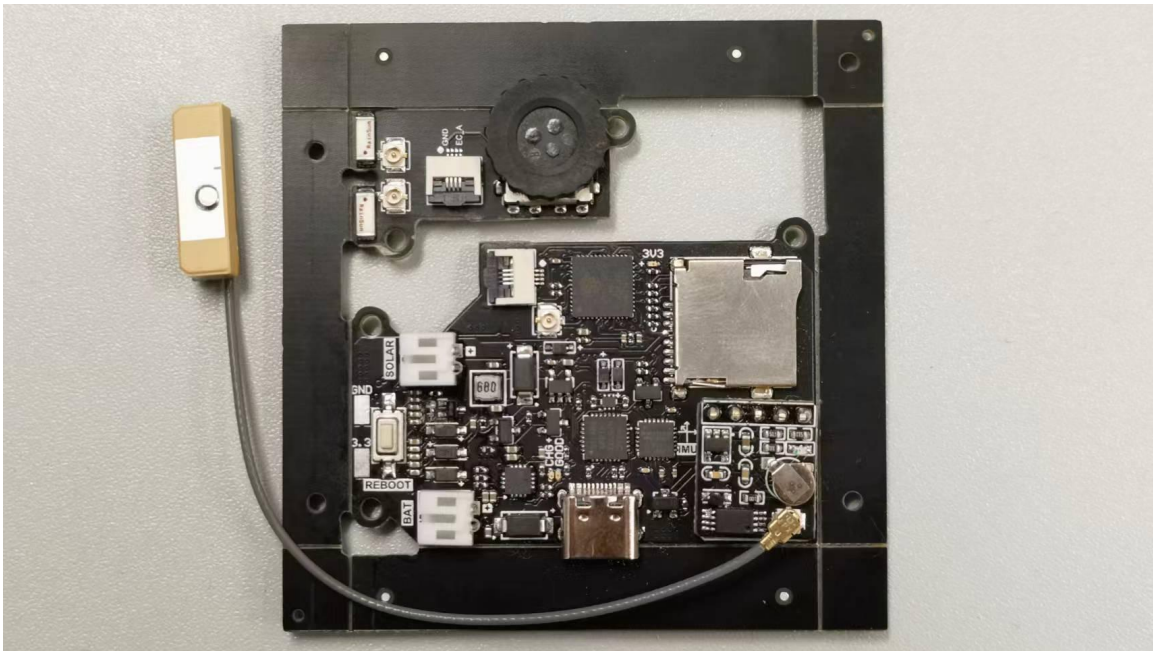
Item	Item Description	Price	Tax	Shipping	Total Cost
2.13inch E-ink	2.13inch E-Ink Display HAT V3 Version, 250x122 Resolution 3.3V/5V Black White TwoColor e-paper Screen Module for Raspberry Pi Zero/Zero W/Zero WH/2B/3B/3B+/4B, Supports Partial Refresh, SPI Interface	$42.64	$2.56	$0.00	$45.20

ESP32-PICO-KIT Development Board	EC Buying ESP32-PICO-KIT Development Board, Integrated ESP32-PICO-D4 Module Builtin ESP32 Chip Embedded 2.4 GHz WiFi & Bluetooth Dual Module 32-Bit Microprocessor Mini Development Board I2C SPI	$37.34	$2.24	$0.00	$39.58
Adafruit Universal USB / DC / Solar Lithium Ion/Polymer charger		$29.90	$5.74	$13.43	$101.47
USB / DC / Solar Lithium Ion/Polymer charger		$17.50	$0.00	$0.00	
Small 6V 1W Solar Panel		$19.95	$0.00	$0.00	
Lithium Ion Polymer Battery - 3.7v 2500mAh		$14.95	$0.00	$0.00	
PCB&SMT	5 PCB and their components, assembly. A 10$ coupon is applied	$285.14	0	$19.05	$294.19
ATGM336H GPS	ATGM336H GPS module	$35.64	$2.13	$0.00	$37.77
miscs	screws, batteries, solar cells, hex nuts, cables	$129.76	$7.79	$6.99	$144.54
cables	I-PEX MHF1 to I-PEX MHF1 cables	$25.55	$1.53	$16.50	$43.58
Einks	Eink screens and its dev-board	$61.96	$0.00	$30.00	$91.96
miscs	screens, SD card adapter, LCD screens	$48.32	$2.91	$0.00	$51.23
3D printing case	Using short code to pay	$19.80	$0.00	$0.00	$19.80
PCB&SMT		$176.75	$0.00	$19.05	$185.80
extend board for eink	extend board for eink. The design has some error so the cable is too short to connect	$0.95	$0.57	$6.99	$17.06
bike computer mount	mount, adapter for the bike computers	$37.97	$2.28	$0.00	$40.25
Einks	1.54inch Eink screens	$7.99	$0.00	$30.00	$61.96
3D printing case	Using short code to pay	$19.80	$0.00	$0.00	$19.80
ph2.0 connector		$8.99	$0.54	$0.00	$9.53
FPC cable	FPC extend cable	$6.99	$0.42	$0.00	$7.41
resistors	resistors used to fix hardware design fause	$7.77	$0.47	$6.99	$15.23
3D printing case	Final version of 3D printing case. A 10$ coupon is applied	$30.92	$0.00	$19.16	$40.08
chips for sensors board	ESP32 pico d4	$38.12	$2.29	$6.99	$47.40
Stencil for chip	Stencil for QFN48 chip soldering	$11.59	$0.70	$0.00	$12.29

The total expense is $1326.13

VIII. References and Citations

PCB design



Type Definitions, Defines, and Global Variables

We have too much software and many of them are open-source projects or drivers. Here introduce some frequently used modules.

We start from the APP layer.

Page: the page is a virtual class and it follows the MVC structure. Take the LiveMap page as an example, the LiveMap work as the controller, LiveMapView work as the view, and LiveMapModel work as the model.

```
namespace Page
{


class LiveMap : public PageBase
{
public:
    LiveMap();
    virtual ~LiveMap();


    virtual void onCustomAttrConfig();
    virtual void onViewLoad();
    virtual void onViewDidLoad();
    virtual void onViewWillAppear();
    virtual void onViewDidAppear();
    virtual void onViewWillDisappear();
    virtual void onViewDidDisappear();
    virtual void onViewDidUnload();


private:
    LiveMapView View;
    LiveMapModel Model;


    static char routeDirPath[ROUTE_DIR_PATH_MAX];


    struct
    {
        uint32_t lastMapUpdateTime;
        uint32_t lastContShowTime;
        uint32_t lastPressedTime;
        uint32_t lastClickededTime;
        bool clickedBefore;
```

```
        lv_timer_t* timer;
        TileConv::Point_t lastTileContOriPoint;
        bool isTrackActive;
        bool isRouteActive;
    } priv;

    static uint16_t mapLevelCurrent;

private:
    typedef  TrackLineFilter::Area_t Area_t;

private:
    void Update();
    void UpdateDelay(uint32_t ms);
    void CheckPosition();
    void Eink_Update();

    /* SportInfo */
    void SportInfoUpdate();

    /* MapTileCont */
    bool GetIsMapTileContChanged();
    void onMapTileContRefresh(const Area_t* area, int32_t x, int32_t y);
    void MapTileContUpdate(int32_t mapX, int32_t mapY, float course);
    void MapTileContReload();

    /* TrackLine */
    void TrackLineReload(const Area_t* area, int32_t x, int32_t y);
    void TrackLineAppend(int32_t x, int32_t y);
    void TrackLineAppendToEnd(int32_t x, int32_t y);
    static void onTrackLineEvent(TrackLineFilter* filter,
TrackLineFilter::Event_t* event);
    /* RouteLine */
    void RouteLineReload();
```

```
    void RouteLineAppend(lv_poly_line* target,int32_t x, int32_t y);
    void RouteLineAppendToEnd(lv_poly_line* target,int32_t x, int32_t y);
    void AttachEvent(lv_obj_t* obj);
    static void onEvent(lv_event_t* event);
    void onBtnClicked(lv_obj_t* btn);
};


}
```

```
namespace Page
{


class LiveMapView
{
public:
    struct
    {
        lv_obj_t* loadingLabelInfo;


        struct{
            lv_obj_t* cont;
            lv_obj_t* info;
        } toEinkLabelInfo;


        lv_style_t styleCont;
        lv_style_t styleLabel;
        lv_style_t styleLineTrack;
        lv_style_t styleLineRoute;


        struct
        {
            lv_obj_t* cont;
```

```
        lv_obj_t* imgArrow;
        lv_obj_t** imgTiles;
        uint32_t tileNum;
    } map;

    struct
    {
        lv_obj_t* cont;
        lv_poly_line* lineTrack;
        lv_obj_t* lineActive;
        lv_point_t pointActive[2];
    } track;

    struct
    {
        lv_obj_t* cont;
        lv_poly_line* lineRoute;
        lv_obj_t* lineActive;
        lv_point_t pointActive[2];
        std::vector<std::pair<double,double>> routePoints;
    } route;

    struct
    {
        lv_obj_t* cont;
        lv_obj_t* labelInfo;
        lv_obj_t* slider;
    } zoom;

    struct
    {
        lv_obj_t* cont;
    } move;
```

```
        struct
        {
            lv_obj_t* cont;

            lv_obj_t* labelSpeed;
            lv_obj_t* labelTrip;
            lv_obj_t* labelTime;
        } sportInfo;
        lv_obj_t* cont;
    } ui;
    struct{
        struct{
            lv_obj_t* Info_GPS; //already in lcd
            lv_obj_t* Data_GPS;
        }gpsInfo;

        struct{
            lv_obj_t* Info_North; //already in lcd
            lv_obj_t* Data_North;
        }northInfo;

        struct{
            lv_obj_t* Info_East; //already in lcd
            lv_obj_t* Data_East;
        }eastInfo;

        struct{
            lv_obj_t* Info_Dest; //already in lcd
            lv_obj_t* Data_Dest;
        }destInfo;

        struct{
```

```
            lv_obj_t* Info_Batt; //already in lcd
            lv_obj_t* Data_Batt;
        }battInfo;

        struct
        {
            lv_obj_t* cont;
            lv_poly_line* lineRoute;
            lv_obj_t* lineActive;
            lv_point_t pointActive[2];
            // std::vector<std::pair<double,double>> routePoints;
        } route;
        lv_obj_t* imgArrow;
        lv_obj_t* cont;

    } eink_ui;

    void Eink_info_init(void);
    void Eink_Update();
    void Create(lv_obj_t* root, uint32_t tileNum);
    void Delete();
    void SetImgArrowStatus(lv_coord_t x, lv_coord_t y, float angle)
    {
        lv_obj_t* img = ui.map.imgArrow;
        lv_obj_set_pos(img, x, y);
        lv_img_set_angle(img, int16_t(angle * 10));
    }
    void SetMapTile(uint32_t tileSize, uint32_t widthCnt);
    void SetMapTileSrc(uint32_t index, const char* src);
    void SetArrowTheme(const char* theme);
    void SetLineActivePoint(lv_coord_t x, lv_coord_t y);

private:
```

```
    void Style_Create();
    void Eink_Item_Create(
    lv_obj_t* Info,
    lv_obj_t* &data,
    const char* infos,
    int x_bias,
    int y_bias,
    bool is_large_font = true
    );
    void Map_Create(lv_obj_t* par, uint32_t tileNum);
    void ZoomCtrl_Create(lv_obj_t* par);
    void SportInfo_Create(lv_obj_t* par);
    lv_obj_t* ImgLabel_Create(lv_obj_t* par, const void* img_src, lv_coord_t
x_ofs, lv_coord_t y_ofs);
    void Track_Create(lv_obj_t* par);
    void Route_Create(lv_obj_t* par);
    void Eink_Route_Create(lv_obj_t* par);
    // static StorageService storageService;
};


}
```

```
namespace Page
{


class LiveMapModel
{
public:
    LiveMapModel();
    ~LiveMapModel() {}
    void Init();
    void Deinit();
```

```
    // bool InitRouteFile(std::string FilePath);
    void GetGPS_Info(HAL::GPS_Info_t* info);
    void GetPower_Info(HAL::Power_Info_t *info);
    void GetArrowTheme(char* buf, uint32_t size);
    bool GetTrackFilterActive();
    void TrackReload(TrackPointFilter::Callback_t callback, void* userData);
    void SetStatusBarStyle(DataProc::StatusBar_Style_t style);

public:
    // std::shared_ptr<FileWrapper> routeFile_=nullptr;
    // FileWrapper routeFile;
    Routes routes;
    MapConv mapConv;
    TileConv tileConv;
    TrackPointFilter pointFilter;
    TrackLineFilter lineFilter;
    HAL::SportStatus_Info_t sportStatusInfo;

private:
    Account* account;

private:
    static int onEvent(Account* account, Account::EventParam_t* param);

// private:
};

}
```

Then we comes to the System Framwork layer.

The data buffer can record data and provide it for pages.

```
static StorageService storageService;
static int16_t RouteGetRange(const char* dirName, std::vector<std::string>*
found);// check route availability
```

```
static bool MapConvGetRange(const char* dirName, int16_t* min, int16_t* max); // check map availability
static bool onLoad(Account* account);// called by onNotify when system start up. load routes and map
static void onNotify(Account* account, Storage_Info_t* info);// called by onEvent
static int onEvent(Account* account, Account::EventParam_t* param);// event callback function.
static void onSDEvent(bool insert);
DATA_PROC_INIT_DEF(Storage);//a macro use to init this buffer
```

There are many utilities in System Framwork layer.

Here is a routes class that record routes information

```
class Routes{
public:
    Routes(){

        PM_LOG_DEBUG("create Routes object");
    }
    ~Routes() {}
    static void SetRouteNum(int16_t num);
    static int16_t GetRouteNum();
    //return the chosen file name in std::string
    static std::string GetRouteChooseName();
    // choose the route that used to navigate by its name
    static void ChooseRouteByName(std::string chooseName);
    // choose the route that used to navigate by its index in vector
    static void ChooseRouteByIdx(int16_t idx);
    static void SetAvailableRoutes(std::vector<std::string> foundRouteName);
    static std::vector<std::string>* GetRouteNamePtr();



private:
    static int16_t numRoute;
    static std::vector<std::string> availableRoutes;

    static std::string routeChoose;
};
```

Then we come to the HAL layer.

Here I give display initialization as an example:

```
typedef TFT_eSPI SCREEN_CLASS;
extern TFT_eSPI tft; /* TFT instance */

extern lv_indev_t* touch_indev;
extern lv_indev_t* encoder_indev;

// #define CS_PIN          9
#define TRANSFER_TO_EINK  digitalWrite(TFT_CS,HIGH);
#define TRANSFER_TO_LCD   digitalWrite(TFT_CS,LOW);

#define TURN_ON_LCD    digitalWrite(CONFIG_SCREEN_PWR_PIN,LOW);
#define TURN_OFF_LCD   digitalWrite(CONFIG_SCREEN_PWR_PIN,HIGH);

/
#define CALIBRATION_FILE "/TouchCalData1"

#define REPEAT_CAL false
// typedef TFT_eSPI SCREEN_CLASS;
void To_LCD_Port();// move to LCD
void To_Eink_Port(); //move to Eink
void LCD_Power_On();
void LCD_Power_Off();
void Port_Init_Eink();
void Port_Init();
void End_spi_transaction();
void startFreeRtos();
void DisplayFault_Init(SCREEN_CLASS* scr);
void lv_port_disp_lcd_init(SCREEN_CLASS* scr);
void lv_port_disp_eink_init();
void lv_fs_if_init();
void lv_port_indev_init();
```

```
extern TaskHandle_t handleTaskLvgl;
```

file that only tackle display driver

```
#define DISP_HOR_RES         CONFIG_SCREEN_HOR_RES
#define DISP_VER_RES         CONFIG_SCREEN_VER_RES
#define DISP_BUF_SIZE        CONFIG_SCREEN_BUFFER_SIZE
#define EINK_DISP_HOR_RES EPD_HEIGHT
#define EINK_DISP_VER_RES EPD_WIDTH
lv_color_t* lv_disp_buf_p;

extern lv_disp_t * disp_lcd;
extern lv_disp_t * disp_eink;

static lv_disp_draw_buf_t disp_buf;
// static lv_disp_drv_t disp_drv;

Epd epdControl;
Paint einkImg(IMAGE_DATA, EPD_WIDTH, EPD_HEIGHT);

static void disp_flush_cb(lv_disp_drv_t* disp, const lv_area_t* area,
lv_color_t* color_p); //LCD flush call back
void set_pixel(int32_t x, int32_t y, lv_color_t color_p);//convert pixel to
eink 2-bit format
void my_disp_flush_eink(lv_disp_drv_t *disp, const lv_area_t *area, lv_color_t
*color_p);//Eink flush call back
static void disp_wait_cb(lv_disp_drv_t* disp_drv);
void lv_port_disp_lcd_init(SCREEN_CLASS* scr);//initialize driver for LVGL
void lv_port_disp_eink_init();
```

file that only tackle input device driver

```

static void encoder_init(void);
static void encoder_read(lv_indev_drv_t* indev_drv, lv_indev_data_t* data);

```

```
lv_indev_t* encoder_indev;
lv_indev_t* touch_indev;
static TouchPoint_t coordinate;
/*Read the touchpad*///choose from quick prototype to M-Track
#ifdef TOUCH_CS
static void my_touchpad_read( lv_indev_drv_t * indev_driver, lv_indev_data_t *
data );
#else
static void my_touchpad_read(lv_indev_drv_t * indev_drv, lv_indev_data_t *
data);
#endif

void lv_port_indev_init(void);

/* Initialize your keypad */
static void encoder_init(void);

/* Will be called by the library to read the encoder */
static void encoder_read(lv_indev_drv_t * indev_drv, lv_indev_data_t * data);
```

file that only tackle file system driver

```
#define SD_LETTER '/'
typedef FIL file_t;

typedef FF_DIR dir_t;
static void fs_init(void);

static bool fs_ready(lv_fs_drv_t* drv);
static void* fs_open(lv_fs_drv_t* drv, const char* path, lv_fs_mode_t mode);
static lv_fs_res_t fs_close(lv_fs_drv_t* drv, void* file_p);
static lv_fs_res_t fs_read(lv_fs_drv_t* drv, void* file_p, void* buf, uint32_t
btr, uint32_t* br);
static lv_fs_res_t fs_write(lv_fs_drv_t* drv, void* file_p, const void* buf,
uint32_t btw, uint32_t* bw);
```

```
static lv_fs_res_t fs_seek(lv_fs_drv_t* drv, void* file_p, uint32_t pos,
lv_fs_whence_t whence);
static lv_fs_res_t fs_size(lv_fs_drv_t* drv, void* file_p, uint32_t* size_p);
static lv_fs_res_t fs_tell(lv_fs_drv_t* drv, void* file_p, uint32_t* pos_p);
static lv_fs_res_t fs_remove(lv_fs_drv_t* drv, const char* path);
static lv_fs_res_t fs_trunc(lv_fs_drv_t* drv, void* file_p);
static lv_fs_res_t fs_rename(lv_fs_drv_t* drv, const char* oldname, const char*
newname);
static lv_fs_res_t fs_free(lv_fs_drv_t* drv, uint32_t* total_p, uint32_t*
free_p);
static void* fs_dir_open(lv_fs_drv_t* drv, const char* path);
static lv_fs_res_t fs_dir_read(lv_fs_drv_t* drv, void* dir_p, char* fn);
static lv_fs_res_t fs_dir_close(lv_fs_drv_t* drv, void* dir_p);

static bool fs_ready(lv_fs_drv_t* drv)
static lv_fs_drv_t fs_drv;   /*A driver descriptor*/

void lv_fs_if_init();
```

common config

```
#define CONFIG_SYSTEM_SAVE_FILE_PATH          "/SystemSave.json"
#define CONFIG_SYSTEM_SAVE_FILE_BACKUP_PATH   "/.SystemSaveBackup.json"
#define CONFIG_SYSTEM_LANGUAGE_DEFAULT        "en-GB"//{'e','n','-','G','B'}
#define CONFIG_SYSTEM_TIME_ZONE_DEFAULT      (-5) // GMT- 5, Ann Arbor
#define CONFIG_SYSTEM_SOUND_ENABLE_DEFAULT    false

#define CONFIG_WEIGHT_DEFAULT                 70 // kg

#define CONFIG_GPS_REFR_PERIOD                1000 // ms
#define CONFIG_GPS_LONGITUDE_DEFAULT          -83.7161   //Ann Arbor duder
#define CONFIG_GPS_LATITUDE_DEFAULT           42.2905
```

```
#define CONFIG_GPS_LONGTITUDE_OFFSET          -0.041925
#define CONFIG_GPS_LATITUDE_OFFSET            0.002818

#define CONFIG_MAP_DOUBLE_CLICKED_DELAY       700
#define CONFIG_MAP_LONG_PRESSED_TIME          1000

#define CONFIG_TRACK_FILTER_OFFSET_THRESHOLD  2 // pixel
#define CONFIG_TRACK_RECORD_FILE_DIR_NAME     "Track"

#define CONFIG_ROUTE_FILE_DIR_NAME            "/Route"

#define CONFIG_MAP_USE_WGS84_DEFAULT          false

#define CONFIG_MAP_DIR_PATH_DEFAULT           "/MAP"
#define CONFIG_MAP_EXT_NAME_DEFAULT           "bin"
#define CONFIG_ARROW_THEME_DEFAULT            "default"

#define CONFIG_LIVE_MAP_LEVEL_DEFAULT         16
#define CONFIG_LIVE_MAP_VIEW_WIDTH            LV_HOR_RES
#define CONFIG_LIVE_MAP_VIEW_HEIGHT           LV_VER_RES

// color config
#define UMICH_COLOR_MAIZE                     lv_color_hex(0xffcb05)
#define UMICH_COLOR_BLUE                      lv_color_hex(0x00274c)
#define UMICH_COLOR_WHITE                     lv_color_hex(0xffffff)

#define COLOR_BACKGROUND                      UMICH_COLOR_BLUE
#define COLOR_FOCUS                           UMICH_COLOR_MAIZE

#define COLOR_TEXT                            UMICH_COLOR_WHITE
#define COLOR_DIM_TEXT                        lv_color_hex(0x939393)
#define COLOR_UNFOCUS                         COLOR_DIM_TEXT
#define COLOR_PRESSED                         COLOR_DIM_TEXT
```

```
/* Simulator */
#define CONFIG_TRACK_VIRTUAL_GPX_FILE_PATH    "S:/TRK_20210801_203324.gpx"

/*=========================
   Hardware Configuration
 *=========================*/

/* Sensors */
#define CONFIG_SENSOR_ENABLE        1

#if CONFIG_SENSOR_ENABLE
#  define CONFIG_SENSOR_IMU_ENABLE  1
#  define CONFIG_SENSOR_MAG_ENABLE  1
#endif

#define NULL_PIN                    PD0

/* Screen */
#define CONFIG_SCREEN_CS_PIN        9
#define CONFIG_SCREEN_DC_PIN        2
#define CONFIG_SCREEN_RST_PIN       4
#define CONFIG_SCREEN_SCK_PIN       18
#define CONFIG_SCREEN_MOSI_PIN      23
#define CONFIG_SCREEN_BLK_PIN       12

#define CONFIG_SCREEN_HOR_RES       240//240
#define CONFIG_SCREEN_VER_RES       320
#define CONFIG_SCREEN_BUFFER_SIZE   (CONFIG_SCREEN_HOR_RES *
CONFIG_SCREEN_VER_RES /2)

/* Battery */
#define CONFIG_BAT_DET_PIN          37
#define CONFIG_BAT_CHG_DET_PIN      38
```

```
/* Buzzer */
#define CONFIG_BUZZ_PIN             25
#define CONFIG_BUZZ_CHANNEL         2
#define CONFIG_SOUND_ENABLE_DEFAULT false

/* GPS */
#define CONFIG_GPS_SERIAL           Serial2
#define CONFIG_GPS_USE_TRANSPARENT  0
#define CONFIG_GPS_BUF_OVERLOAD_CHK 0
#define CONFIG_GPS_TX_PIN           36
#define CONFIG_GPS_RX_PIN           19

/* IMU */
#define CONFIG_IMU_INT1_PIN         32
#define CONFIG_IMU_INT2_PIN         33

/* I2C */
#define CONFIG_MCU_SDA_PIN          32
#define CONFIG_MCU_SCL_PIN          33

/* Encoder */
#define CONFIG_ENCODER_B_PIN        34
#define CONFIG_ENCODER_A_PIN        35
#define CONFIG_ENCODER_PUSH_PIN     27

/* Power */
#define CONFIG_POWER_EN_PIN         21
#define CONFIG_IMU_GPS_PWR_PIN      25
#define CONFIG_SCREEN_PWR_PIN       5

/* Debug USART */
#define CONFIG_DEBUG_SERIAL         Serial
```

```
/* SD CARD */
#define CONFIG_SD_SPI                HSPI
#define CONFIG_SD_CD_PIN             -1
#define CONFIG_SD_CS_PIN             15
#define CONFIG_SD_DET_PIN            22

/* Stack Info */
#define CONFIG_USE_STACK_INFO        0
```

log macro that work for both simulator and real hardware

```
#if !defined(ARDUINO) && DATA_CENTER_USE_LOG
#include <stdio.h>
// #  define _DC_LOG(format, ...)      printf("\r\n[DC]"format, ##__VA_ARGS__)
#  define DC_LOG_INFO(format, ...)  printf("\r\n[DC INFO] "),printf(format,
##__VA_ARGS__)
#  define DC_LOG_WARN(format, ...)  printf("\r\n[DC WARN] "),printf(format,
##__VA_ARGS__)
#  define DC_LOG_ERROR(format, ...) printf("\r\n[DC ERROR] "),printf(format,
##__VA_ARGS__)
#  define DC_LOG_USER(format, ...) printf("\r\n[DC User] "),printf(format,
##__VA_ARGS__)
#else
#include "Arduino.h"
#  define DC_LOG_INFO(format, ...) Serial.printf("\r\nDC INFO:
"),Serial.printf(format, ##__VA_ARGS__)
#  define DC_LOG_WARN(format, ...) Serial.printf("\r\nDC WARN:
"),Serial.printf(format, ##__VA_ARGS__)
#  define DC_LOG_ERROR(format, ...) Serial.printf("\r\nDC ERROR:
"),Serial.printf(format, ##__VA_ARGS__)
#  define DC_LOG_USER(format, ...) Serial.printf("\r\nDC USER:
"),Serial.printf(format, ##__VA_ARGS__)
```

Source Code Access

Our source code can be found in this repository:

https://github.com/Flanker-E/All_In_One_Outdoor_Sports_Comupter

Referenced code: X-Track:

https://github.com/FASTSHIFT/X-TRACK

Peak:

https://github.com/peng-zhihui/Peak

Peak-T2:

https://gitee.com/forairaaaaa/peak-t2