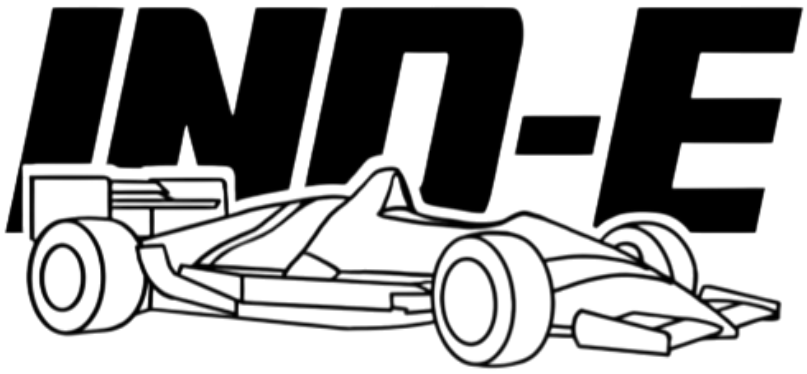
Ind-E Car Electrical Conversion – Control System: HMI Manual



By: Jonathan Mira-Acosta

Date: May 5, 2025

Revision: 1a

Table of Contents

Table of Tables

Table of Figures

Introduction

The Ind-E Human Machine Interface (HMI) is a display developed for the Ind-E conversion capstone project. It was one aspect of the Ind-E Control system. This file doesn't cover any of the communication to other devices through the CAN Bus system. Though it will be mentioned throughout the manual, the only thing to know about CAN Bus is that it's the communication system.

The HMI purpose is to record and display all the data that is being recorded. This is to ensure safe operation of the vehicle and to allow further analysis of its operation. It was designed to be used on a 7-inch touch display, primarily using touch controls. And only using a keyboard for configuration.

The purpose of this manual is to be both a user guide and programmer guide for the HMI. Additionally, it is for anyone who wants to learn more about the coding process or learning more about Python or Kivy. But it doesn't go into installing both Python or Kivy.

To fully understand the manual, a basic understanding of coding would be beneficial. However not needed, since the point is to learn the foundation of the language.

Layout

Due to the amount of data to be displayed, it was decided that the HMI would have multiple screens. Each screen is built from the same starting layout. This allows universal data and functions to be located at the same positions.

The following table shows a summary of all the pages.

Table 1: HMI Page Summary

Page Name	Summary
Universal Elements (Not an actual page)	Data and function that would be universal for all screens. This would be safety indicators, recording indicator, buttons, date and time, and vehicle speed.
Home	Displays only relevant data when driving the vehicle. This would include only data that is relevant to driving.
Battery	Contains detail information on the battery. This would be temperature, charge, current, and BMS functionality.
Motor	Contains detail information on the motor. This would be speed, current, torque, and temperature.
Miscellaneous (Misc)	This is a blank screen to display other information in the future.
Setting	Mostly blank screen to put configurable settings in the future. Currently able to change the recording file name from this page.

The pages are connected in a circular layout, in the order as seen in Table 1. This means by starting from the Home page and going to the next or right page, the order would be Home → Battery → Motor → Misc → Settings → Home.

Universal Elements

Figure 1 shows the universal elemnts used in all pages. Since the Misc page was left blank, it was used to show.

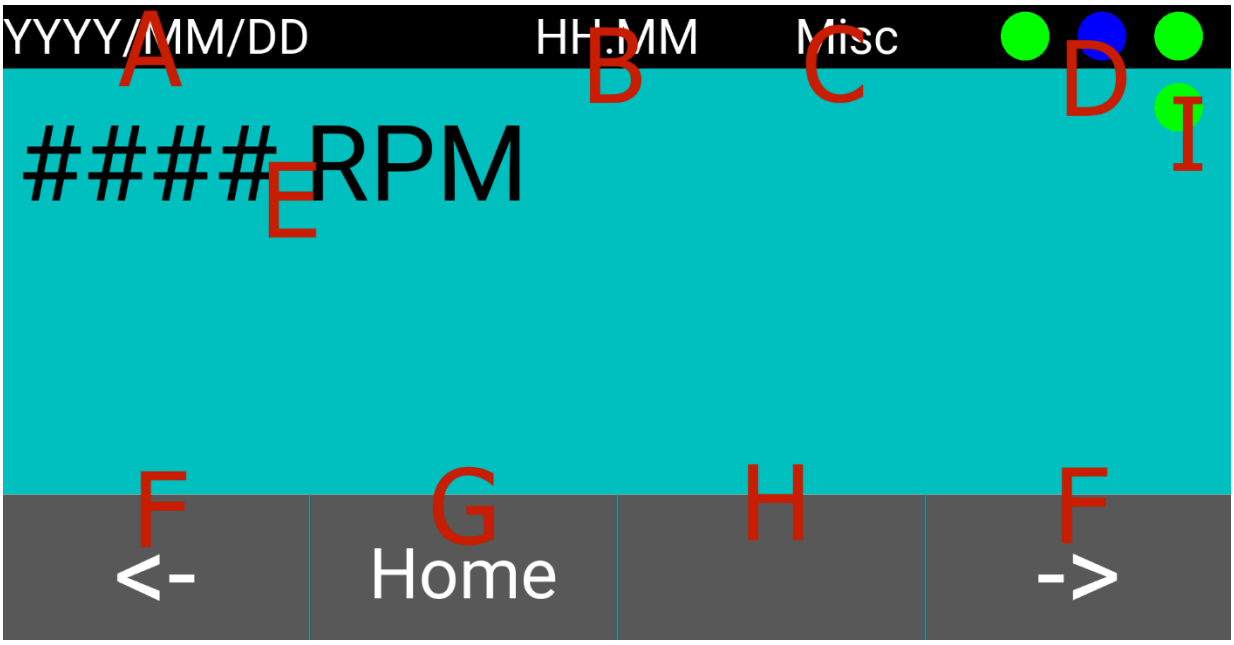


Figure 1: Universal Elements of HMI Pages

A) Date in year/month/day format.
B) Time in 24-hour format, displayed as hour:minute.
C) Displayed page.
D) 3 overall safety indicators: Temperature (left), Battery Charge (Middle), Emergency Stop (right). If any of these indicators are red (Warning), operation of the vehicle should stop and the issue should be investigated.
   - Temperature: Each temperature measurement has 2 boundaries for “Caution” and “Warning”. These bound are set for temperature above the nominal operating temperatures and are different depending on what the measurement is for. This is indicator shows the worst state of all the temperatures. Green if all temperatures are within nominal operating temperature. Yellow if any temperature is past the “Caution” bound. And red if any temperature is past the “Warning” bound.
   - Battery Charge: Indicates the current charge of the batteries. It contains 2 bounds for “Caution” (Yellow) at 50%, and “Warning” (Red) at 25% charge. If it indicates blue, then at least 1 battery charge measurement has not been received by the HMI.

- Emergency Stop (E-Stop): Indicates if the E-Stop switch has been pressed. Would turn red if it has. However, due to the current implementation of the E-Stop, pressing the E-Stop shuts down the whole vehicle. Shutting of the HMI.

E) Rotation of the motor. Can be changed to show the speed of the vehicle in km/h.
F) Page navigation buttons. Pressed to change the pages.
G) Auxiliary button A. Button that changes function depending on the page. For pages not next to the Home page, become a home button. Which changes the page to the Home page.
H) Auxiliary button B. Button that changes function depending on the page.
I) Recording indicator. A green circle that is visible when recording.

Home

The Home page was designed to only display relevant data when driving. Figure 2 shows that added elements.

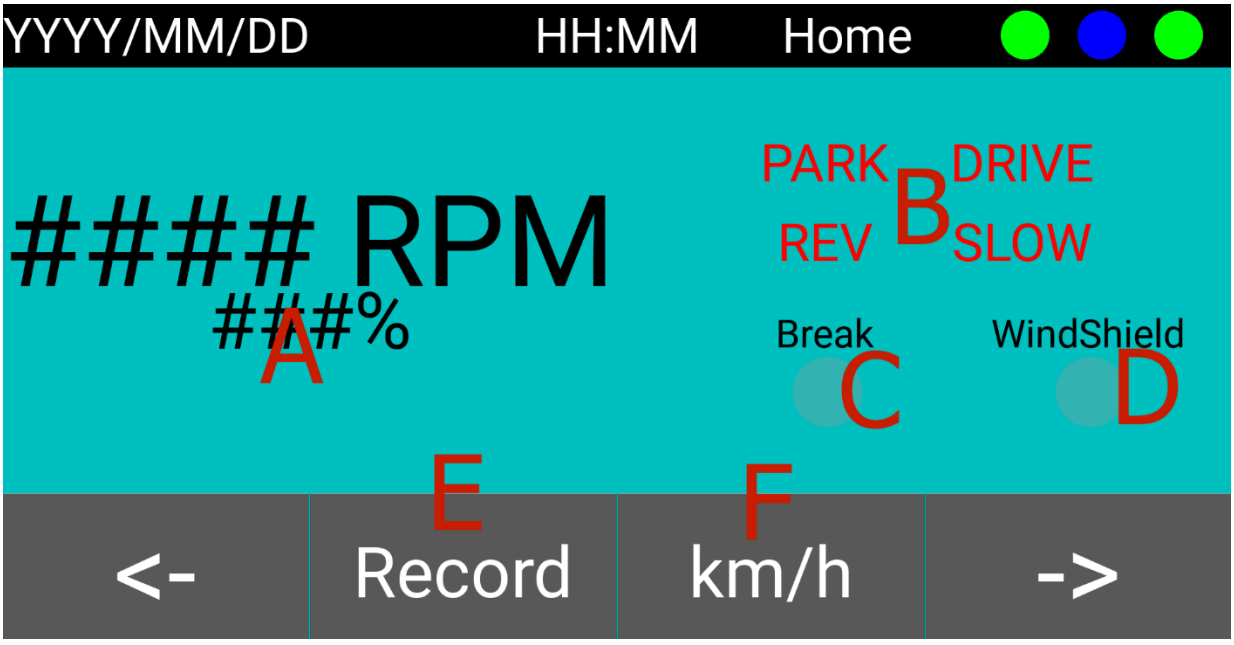


Figure 2: Home Page Layout

A) Throttle position in percentage.
B) Drive mode. Current drive mode will be black while the others are white. Turns red when non are selected.
C) Brake indicator. Turns the circle green when the breaks are pressed.
D) Wind shield wiper indicator. Turns the circle green when the wind shield wipers are activated. Current there are no wipers on the vehicle.
E) Record buttons. Starts and stops data recording. Recoding is saved in a separate folder named “IndE_Data”. Which is located with the project folder. If no such folder exists, it will automatically create it.

F) RPM ↔ km/h toggle. Changes the speed display between RPM and km/h. The button label changes to the non-displayed data. This change applies to the other pages as well.

Battery

Displays important data for the batteries. Due to the amount of measurement for the batteries, not all measurements are displayed. Figure 3 shows the Battery page.

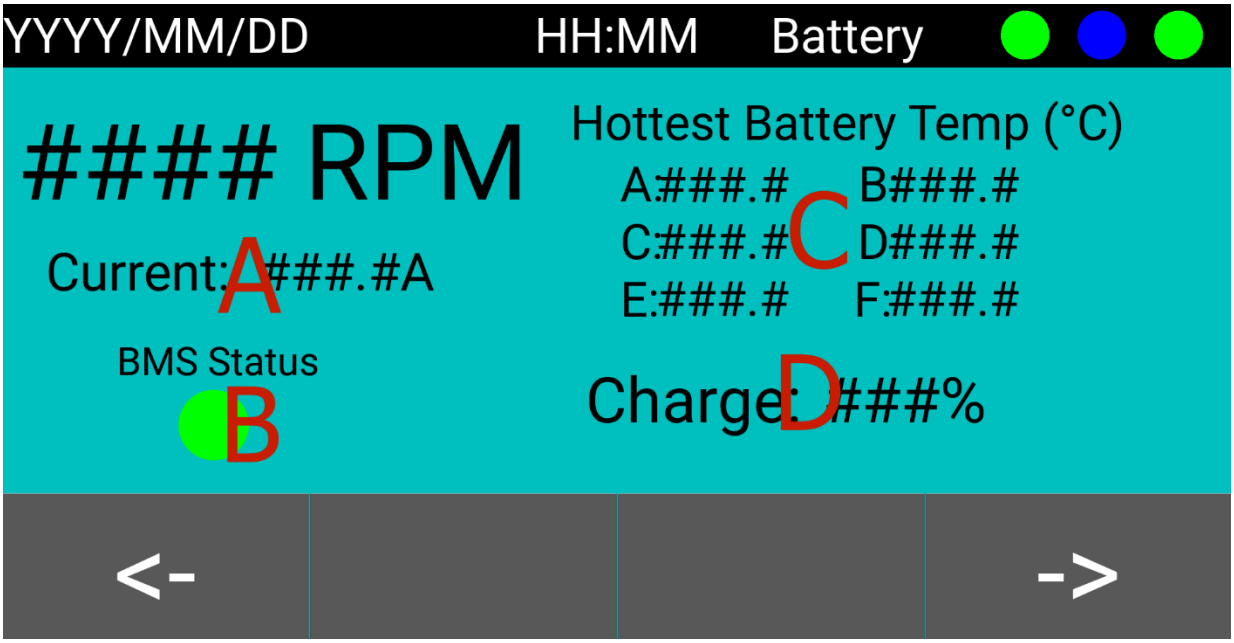


Figure 3: Battery Page Layout

A) Current draw from the batteries.
B) BMS status. Indicates if the BMS is working properly.
C) Hottest battery temperature. Displays the hottest measured temperature for each battery. There are 4 temperature measurements for each battery.
D) Charge of the batteries.

Motor

This page displays values related to the motor and its power delivery system. The motor is a DC 3 phase motor, meaning it requires 3 separate power supply.

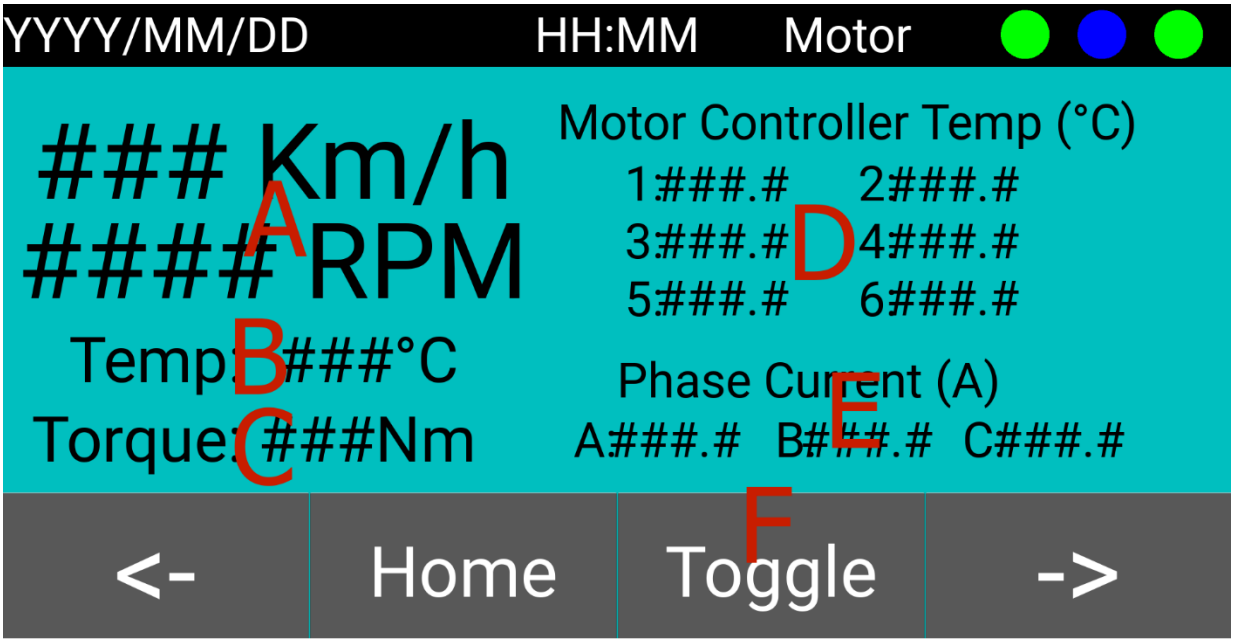


Figure 4: Motor Page Layout

A) Motor rotation speed and vehicle speed. Shows both regardless of what the other pages are set to.
B) Motor temperature.
C) Motor torque.
D) Motor controller temperature. There are 2 for each phase.
E) Phase current.
F) Toggle button. Currently does nothing. Originally would toggle to a different motor display, which would so more information on the different phases. This was removed do to removing some measurements and no longer needing a second display for the data.

Miscellaneous (Misc)

The Misc page is a blank page and can be seen for Figure 1 to display the universal elements. This page was added as a creative space to add anything else that was needed after all the other pages where made. Some ideas for what this page could have been for was sponsor or a team acknowledgement.

Setting

This page was created to hold any of the settings needed. Currently not many settings are available due to a lack of meaning full changes that needs to be made while the HMI is running.

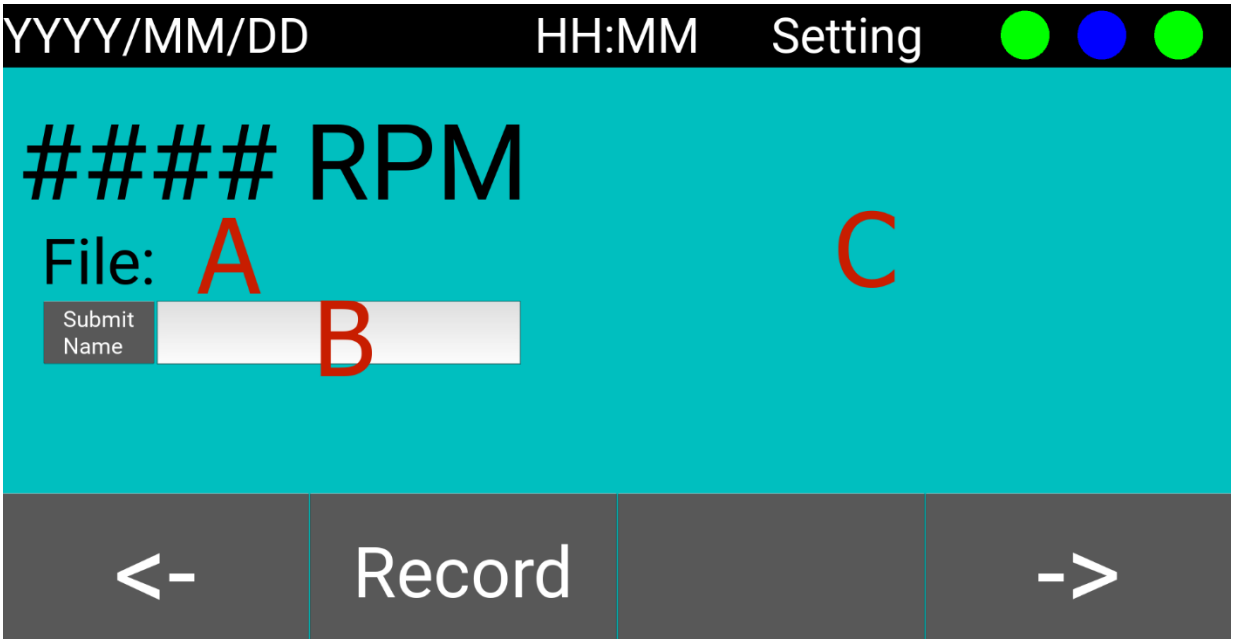


Figure 5: Setting Page Layout

A) File name for recorded data. When recording the data, it will make a Comma-separated values (CSV) file to contain data. The file name consists of the displayed file name followed by the date (YYYY-MM-DD) and time (HH-MM-SS). This format was done so the same description can be used for multiple recording without it over writing past recordings. Making it quicker to do multiple recording one after the other. Note that the file name doesn't save after rebooting the HMI.

B) File name input. This is where the desired file name can be inputted. Ensure that you click the "Submit Name" button or click the 'Enter' key when done. There is not checked to ensure the desired name is valid. To ensure a valid file name, refrain from using the following symbols: '#', '?', '/', '\', '<', '>', '|', and '*'.

C) An empty space to add future settings.

Coding Language Introduction

This section talks about the structure or feel of the coding language. It is less about knowing what needs to be typed and more of how the code is delt with.

Additionally, any features that was considered or researcher but not implemented will also be talked about. The exact code reason for the conflict will not be discussed. To see the code conflict, see the Conflict with Un-used section.

Python

Python is known for being beginner friendly with a large community. The results in it having a large variety of extension that can be used for common projects.

Where python is particular about is formatting. Instead of using symbols to format code, like what some other languages does, it uses the visual structure of the codes. This means that every code statement needs to be on its own line and is grouped based on how indented it is. There are methods to get around this, like using "\" as shown in ***[Insert figure reference]***, but generally it won't be necessary.

Another thing to be aware of is referencing. When coding, it is common to create repeated sections of code separately as its own entity, known as a function or object. Functions are generally an algorithm, think of rolling a die. On the other hand, objects are a collection of functions and variables, this would be the die itself. When creating the separate code for an object, it's like creating a template of the object. This allows you to make as many of those objects as needed, know as an instance. To visualize with the dice, in the code a dice template is made, then you can use that template to make multiple dice instances where you needed them. Each with the same functionality.

Going back to referencing, from what I've researched, when a function is called the code looks for the function, then does said function. This works fine when there's only one function. However, it's more complicated with objects. When creating objects instances, all the functions for the object also get created again. So, when calling one of those functions, the code will go looking for the function and now see multiple of them. Thinking of the dice example, it's like having 5 dice, picking up die #2, then being told to roll dice some dice. The code might roll die #2, but it might also roll any other dice. To add to the compilation, the object template also functions as a usable object. So instead of using one of the object instances, the code might use the object template.

To solve this, "self" is added to the function to reference that particular instance of the objects. So instead of picking up a die out of multiple and being told to roll a die. Instead, the command would be to roll the die that was picked up.

Kivy

Kivy in an extension to Python, that is designed for creating displays. It does this by creating and running an app. Within this app are different blocks that can be configures. Each block having different options to configure.

Along with Python codes, Kivy has its own sudo-coding language. This Kivy code must be interfaced through Python code and doesn't contain any hard logic. The code is contained in a separate file, which makes it resembles a configuration file. In that it's where different elements of the display can be defined and change the appearance. However, any logic, besides basic math, will still be coded in the Python file. Alternatively, the configuration done in the Kivy code can be done in the Python code instead.

The Kivy code follows the feel of Python. Where the formatting of code matters for linking parts of the code. And from experience it's a lot easier to get running. This is due to incorrectly configuring an element won't cause an error. What this means is that changing the radius of a square will result in nothing happening and the program still runs. This also makes it harder to debug errors.

Where Kivy excels at is adaptive organization and simple implementation of interactive elements. Simply put, there is a row of 3 buttons within a fixed space, adding a 4^{th} of 5^{th} is extremely simple. Assume it was originally configured correctly. When adding the additional buttons, Kivy will automatically resize all the buttons to ensure they all fit evenly. And getting the buttons to work, is a matter of setting the correct function to activate.

Un-used Features

Similar to Python, Kivy have multiple features for a variety of projects. Some of these where research and would have been used, if not for some integration issues. Here, these features will be discussed as an overview. To see the coding issues, see the Conflict with Un-used Features section.

Builder

The Builder is used to select the Kivy code file to create the app from. If the Builder is not used, then 1 object will be selected to be the app, and anything that falls under it would get selected with it. This means that if the app has a single main object, then the Builder is optional. However, for more complex programs with multiple independent objects, then the Builder is needed.

An example would be fruit. If the project is 1 apple, them selecting the apple or the peel would be fine. But if it's many fruits, then the entire basket needs to be selected.

Screen Manger

The Screen Manger does what it says. It allows for multiple screens to be created and manages those screens. It allows for screens to be named and to change select based on names. Having

multiple screens is efficient as it allows for information to be divided and organized. And reduces wasted power by only displaying what is needed.

Icon

Icons are useful for providing information and reducing labels in a simple and pretty fashion. To implement icon, KivyMD needs to be used. KivyMD is an extension to Kivy. When using KivyMD, adding icons becomes as simple as adding text or a button.

The difficulty is that it replaces Kivy. This means that the Kivy app will now be a KivyMD app. Since the base is the same, it should work. However, the difference may lead to the reprograming of half of the app.

Code

DataLog.py

IndE_HMI.py

Conflict with Un-used Features

inde_display.kv

Future Additions