NOVA Rocket Telemetry

Goals/Features:

- Collect data from the 3 Microcontrollers
- Save in the SD card
- Send to the ECHO via E32-900T30D
- Show the data with interface of SONIC and CANSAT

SONIC - Communication Computer

Sonic takes the data from FC1, FC2 and Igris (Power Monitor) via I2C Communication. Writes on the SD card and sends to the ECHO via E32-900T30D

Sonic has,

ESP32-S3 microcontroller, ESP32-S3 has dual core so we can distribute the tasks between both the cores

E32-900T30D for the long range communication over air, Probably to be set at 865MHz

SD card for storing the data

Boot and **Enable** buttons

Passive components such as **Decoupling Capacitors**, **Pullups/Pulldown Resistors**

Test Pins/Pads

USB TYPE C for USB to UART Communication

ECHO - Ground Computer

Echo takes the data from the **SONIC** and sends to the **Ground Station PC** for the Web Interface via USB

Echo has,

ESP32-S3 for its dual core optimization. As there are two receivers on the ECHO one for the **SONIC** and another for the **CANSAT** communication (Probably will be at 867MHz)

2 E32-900T30D for communicating with SONIC and CANSAT

Same Passive Components as SONIC

Boot and **Enable** buttons

RGB leds for indicating Signal Strength

USB TYPE C for USB to UART Communication

Test Pins/Pads

CORE FUNCTIONS:

SONIC

SONIC's **core1** is to be utilised to read continuosly data from the **FC1, FC2, IGRIS** on loop one by one

Data from the all three are to be the **CSV format**. Core1 will convert this CSV format into **Binary format** thus it will reduce the packet size up to 50% thus we can get much **higher data rates.**

Core1 then send this data into the Queue thus here **core0** can access the data core0 will write the data on the **SD card** and also sends to **E32** for the Transmission

ECHO

ECHO's **core1** will be **utilised** to read the data from the 1_E32 which is listening to the SONIC's E32 and **core0** will be **utilised** to read the data from the 2_E32.

Both the cores will serial write on the port of the connected PC and when to write will be decided based on whoever gets the data first. **Note** that the ECHO is just transmitting the binary format it gets directly to the PC for the **faster communication** and PC will have an code which will parse the binary format and show us on the **UI Dashboard**, this dashboard runs on a **local webserver**

```
import time
    import random
    import dash
    from dash import dcc, html
    from dash.dependencies import Input, Output
    import plotly.graph_objs as go
```

The above are the libraries to be used for the UI purposes

<u>DATA CONVERSION FLOW:</u>

CSV to Binary

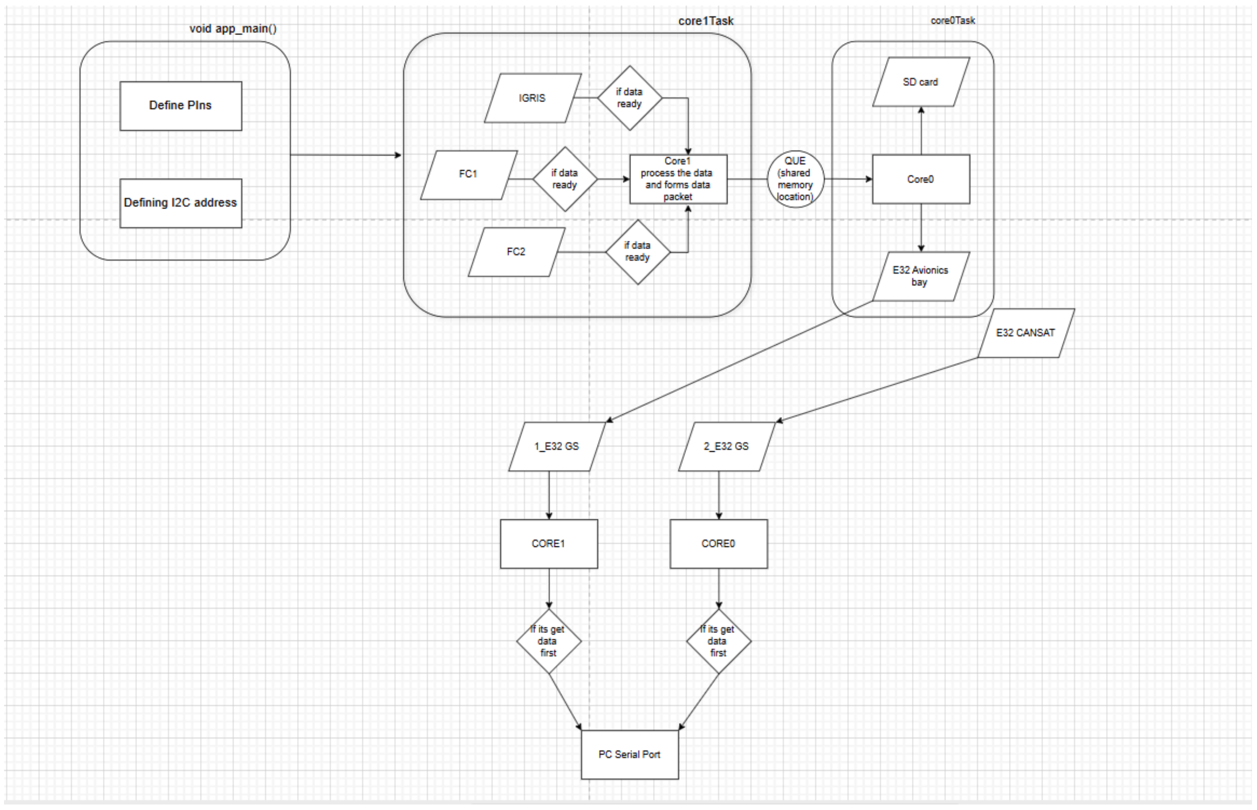
Binary to Readable Format on the Ground station

Example:

CSV : 123, 456, 78.9, 3.7

Binary : 0x007B 0x01C8 0x42 0x4C 0x00 0x00 0x0E 0x94

Readable : Altitude: 123 m
Speed: 456 m/s
Temperature: 78.9 °C
Battery: 3.7 V


void app_main()
Define PIns
Defining I2C address
core1Task
IGRIS
if data ready
FC1
if data ready
Core1 process the data and forms data packet
FC2
if data ready
QUE (shared memory location)
core0Task
SD card
Core0
E32 Avionics bay
E32 CANSAT
1_E32 GS
2_E32 GS
CORE1
CORE0
if its get data first
if its get data first
PC Serial Port