Hello, so regarding the safety node, are you open to really using both sofware(ROS2 node) and hardare(ESP32)?

here's what the microcontroller can do;

|  |  |  |
| --- | --- | --- |
| **Requirement** |  | **ESP32 Capable?** |
| Enforce startup sequence from Beginner’s Guide | Logic-based sequencing | Yes – FSM or state machine in Arduino(in our case) |
| Track ASSI state (AS Off, AS Ready, AS Driving…) | Deterministic FSM | Yes – simple state machine with GPIO/state tracking |
| Gate throttle commands unless in "Active"/"AS Driving" | Real-time signal gating | Yes – control PWM outputs or relay/mosfet gating(we'll see what's available. |
| Monitor inputs like brake status, emergency stops, watchdog | Signal I/O, debounce, edge detection | Yes – digital/analog GPIO + ISR |
| Block/override unsafe commands | Gating + logic decision | Yes – logic-based output control, or override signal line |
| Trigger failsafe conditions (like activating EBS) | Immediate GPIO control | Yes – digital output trigger or latching relay |
| Log or reflect state over communication | Serial/UART or CAN | Yes – via UART, SPI, CAN (with MCP2515 or similar although will be challenging as maker space is close currently) |
| Display(diagnostic/debug) | Optional – status lights / serial | Yes – via LED + OLED --- I have both of these. |

here's what I think would be better to be done in the ROS2 node;

|  |  |
| --- | --- |
| **Potential Issue** | **Explanation** |
| CAN integration | ESP32 has no native CAN – need transceiver chip like MCP2515(we can just delegate this to the node instead of ordering/asking maker space). |
| Watchdog sync with ROS2 | ROS2 have the watchdog |
| Startup sequencing logic | Must follow D2.6 strictly, including timeouts and conditions |

So basically(as you may already know);

**We would build the FSM (Finite State Machine) in the ROS2 node**, which makes all high-level decisions (like AS\_Off → AS\_Ready → AS\_Driving → AS\_Emergency).  
Then the **ESP32 acts like the *enforcer* or *executor*** — it:

* Listens to what state it's in.
* Controls hardware accordingly.
* Monitors critical safety signals *in parallel* and can override if ROS2 fails.