Guide: Adjusting RS485 Communication in diyBMS v4 ESP32

This guide describes the modification of RS485 communication in diyBMS v4 ESP32 to ensure that only the active master communicates with the Current Monitor via RS485.
The existing master detection over CAN bus remains unchanged, and only the RS485 functionality is optimized.

1. Use existing pins for RS485

The existing RS485 pins from diyBMS v4 ESP32 will continue to be used:

```
#define RS485_DE_PIN 16  // RS485 Data Enable
#define RS485_RE_PIN 17  // RS485 Receive Enable
```

2. Function to Control RS485 Mode

This function switches between sending and receiving:

```
void setRS485Mode(bool master) {
    if (master) {
        digitalWrite(RS485_DE_PIN, HIGH); // Enable sending
        digitalWrite(RS485_RE_PIN, HIGH);
    } else {
        digitalWrite(RS485_DE_PIN, LOW); // Enable receiving mode
        digitalWrite(RS485_RE_PIN, LOW);
    }
}
```

3. Ensure only the Master sends

The send function is modified so that only the master sends data:

```
void sendDataToCurrentMonitor(const uint8_t* data, size_t length) {
    if (!isMaster) return; // Only the master sends!

    setRS485Mode(true);  // Switch RS485 to send mode
    delay(5);            // Wait for stability

    Serial2.write(data, length);
```

```
    Serial2.flush();     // Ensure all data is sent

    delay(5);            // Short delay after sending
    setRS485Mode(false); // Switch RS485 back to receive mode
}
```

4. Ensure all devices listen

The Current Monitor response is received by all BMS devices:

```
void read_response() {
    setRS485Mode(false); // Ensure we are in receive mode

    if (Serial2.available()) {
        uint8_t responseData[32];  // Example size
        Serial2.readBytes(responseData, sizeof(responseData));

        // Process the received data
    }
}
```

5. Integrate RS485 functions into loop()

The loop function controls the system behavior:

```
void loop() {
    checkCANBusForMaster();  // Regularly check master status

    if (isMaster) {
         uint8_t requestData[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x06}; // Example Modbus
request
        sendDataToCurrentMonitor(requestData, sizeof(requestData));
    }

    read_response(); // All BMS listen to the response
    delay(1000);
}
```

Conclusion

These adjustments ensure that only the active master communicates with the Current Monitor via RS485, while other devices remain passive.

The existing master detection over CAN bus remains unchanged, and only the RS485 control has been modified.