1kW ???????? (AWD EV) ????????

1. ?????? (System Architecture)

1.1 ???? (Powertrain)

- **電池電源: 13s**鋰電池組, 最大輸出功率限制 **1000W** (約 20.8A)。
- **驅動馬達: 4**顆 x 500W 20吋 輪轂馬達 (總額定功率 2000W, 採降額使用)。
  - 前軸 **(Front Axle):** 內建 單向離合器 **(Overrunning Clutch)**。
    - 特性: 加速時接合, 滑行時空轉無磁阻, 無法回充/倒車。
  - 後軸 **(Rear Axle):** 直驅 (Direct Drive)。
    - 特性: 可雙向控制, 具備 動能回收 **(Regen)** 與 扭矩向量 **(Vectoring)** 能力

Gemini_Generated_Image_y1i7ppy1i7ppy1i7

1.2 ???? (Control Unit)

- **MCU:** STM32F446 (Cortex-M4F, 180MHz)。
- **OS:** FreeRTOS (CMSIS-V2 API)。
- **通訊:** CAN Bus (500kbps) 連接 4 組 VESC 驅動器。
- **感測器:**
  - 油門踏板 (ADC)。
  - 方向盤轉角感測器 (ADC 或 Encoder)。
  - IMU 慣性感測器 (I2C/SPI, MPU6050/ICM20600)。

---

2. ?????? (Control Strategy)

由於電池僅有 1kW 輸出, 而馬達總容量達 2kW, 控制核心在於 **「功率預算分配」與「動能切換」**。

2.1 ??????? (Drive Mode FSM)

- **狀態 A:** 起步與爬坡 **(4WD Mode)**
  - 條件: 車速 < 35 km/h (可調)。

  - 分配：前後輪同時輸出。
  - 比例：建議前 **40% /** 後 **60%** (利用後輪加速時重心後移增加抓地力)。
- 狀態 **B:** 高速巡航 **(RWD Mode)**
  - 條件：車速 > 35 km/h。
  - 分配：前 **0% /** 後 **100%**。
  - 優勢：前輪離合器脫開進入無阻力滑行，1kW 功率全數集中於後輪，並配合弱磁控制 (Field Weakening) 衝擊極速。

2.2 ?????? (Active Dynamics)

- 電子差速 **(Electronic Differential):** 根據方向盤角度，計算阿克曼轉向幾何，調整左右輪速差。
- 扭矩向量 **(Torque Vectoring):** * 當 IMU 偵測轉向不足 (Understeer) 時，增加外側後輪扭力，煞車內側後輪。
  - 限制：前輪因有離合器，計算出的負扭力指令將被強制歸零 (Clamp to 0)。

---

3. ???????? (Detailed Logic Analysis)

本系統架構基於 RTOS 設計，以下為各邏輯區塊的詳細運作機制：

3.1 ????? (Observer Layer)

負責將原始數據轉換成物理意義，並計算當前的「邊界條件」。

- **CalcPower (**功率預算**):** 最重要的守門員。
  - 公式: $Max_Current = 1000W / V_bat$
  - 作用：確保無論後續如何分配扭矩，總電流基數永遠不會讓電池過載。
- **CalcSpeed (**速度計算**):**
  - 邏輯：主要參考後輪 **(Rear Wheels)** 的 ERPM。
  - 原因：前輪具有離合器，放開油門滑行時前輪轉速可能低於車速（空轉），無法代表真實車速。

3.2 ??? (Strategy Layer) - ??

- **ModeLogic (**模式切換**):**
  - **4WD** 模式 **:** 設定 Splitter 比例為前 **33% /** 後 **67%** (符合前 250W/後 500W配置比例)。
  - **RWD** 模式 **:** 設定 Splitter 比例為前 **0% /** 後 **100%**。
- **EDiff (**電子差速**):**
  - 計算轉向時的幾何差。
  - 邏輯：左轉時，右輪（外側）電流增加 ΔI，左輪（內側）電流減少 ΔI。
- **TCS_ESP (**動態穩定**):**
  - 比較「方向盤角度想要的轉向率 (Target Yaw)」與「IMU 測到的實際轉向率 (Actual Yaw)」。

  - 若差距過大 (出現 Understeer/Oversteer)，產生修正扭矩 ΔI_{ESP}。

3.3 ??????? (Mixer Layer) - ?????

這是數學運算最複雜的地方，負責將基礎扭矩與修正量混合：

- 後輪邏輯 **(Rear Logic - 直驅):** $$I_{Rear_L} = (I_{Total} \times Ratio_{Rear} \times 0.5) + \Delta I_{Diff} + \Delta I_{ESP}$$ $$I_{Rear_R} = (I_{Total} \times Ratio_{Rear} \times 0.5) - \Delta I_{Diff} - \Delta I_{ESP}$$
  - 特點：後輪是直驅，可以接受負值（電子煞車/回充）來修正車身動態。
- 前輪邏輯 **(Front Logic - 離合器特殊處理):** $$I_{Front_L} = (I_{Total} \times Ratio_{Front} \times 0.5) + \Delta I_{Diff}$$ $$I_{Front_R} = (I_{Total} \times Ratio_{Front} \times 0.5) - \Delta I_{Diff}$$
  - 重要限制：由於前輪有單向離合器，當 Mixer 算出負值時必須強制歸零 **(Clamp to 0)**。前輪無法產生制動力，只能靠「少出力」或「多出力」來協助。

3.4 ????? (Limiter Layer)

發送給 VESC 之前的最後一道防線：

1. 單體限制：檢查單顆電機是否超過其額定電流 (例如前輪 Max 10A)。
2. 總量限制：再次加總四顆電機的最終指令，確保 $\sum I \le I_{max_battery}$。如果因為 ESP 修正導致總電流超標，則等比例縮小 (Scale Down) 所有指令。

---

4. ????? (Implementation)

???? (Software Architecture)

採用 **FreeRTOS** 多任務系統，確保控制迴圈的即時性。所有運算採用整數 **(int32)** 以優化 STM32 執行效率。

任務名稱	優先級	頻率	功能描述
`Task_Control`	**Realtime**	**1kHz**	核心控制迴圈 (讀取狀態 -> 策略運算 -> CAN 發送)
`Task_CanRx`	High	Event	接收 VESC 回傳封包 (轉速、電流)，更新車輛狀態
`Task_Input`	Normal	100Hz	讀取油門 ADC、方向盤角度、IMU 數據

以下程式碼基於 STM32 HAL 庫與 CMSIS-OS V2。

4.1 ????? (main.h)

```
/* main.h - 參數與結構定義 */
#ifndef __MAIN_H
#define __MAIN_H

#include <stdint.h>

// --- 物理限制 ---
#define POWER_LIMIT_W          1000     // 電池功率限制
#define BATTERY_VOLTAGE_NOM    48       // 標稱電壓
// 最大總電流 (mA) = 1000W / 48V = 20833 mA
#define MAX_TOTAL_CURRENT_MA   ((POWER_LIMIT_W * 1000) / BATTERY_VOLTAGE_NOM)

// --- 控制參數 ---
#define ERPM_THRESHOLD_RWD     10000    // 切換後驅的轉速閾值 (ERPM)
#define ERPM_HYSTERESIS        500      // 遲滯區間
#define STEERING_GAIN          50       // 轉向靈敏度係數 (整數縮放)
#define VECTORING_GAIN         80       // 向量控制強度係數

// --- VESC CAN ID ---
#define VESC_ID_FL             1
#define VESC_ID_FR             2
#define VESC_ID_RL             3
#define VESC_ID_RR             4
#define CAN_PACKET_SET_CURRENT 1

// --- 共享車輛狀態結構 ---
typedef struct {
    int32_t  avg_erpm;         // 平均車速 (ERPM)
    int32_t  throttle_adc;     // 油門 (0-4095)
    int32_t  steering_val;     // 方向盤 (-2048 左 ~ +2048 右)
    int32_t  yaw_rate_imu;     // IMU Z軸角速度
    uint8_t  drive_mode;       // 0: 4WD, 1: RWD
} VehicleState_t;

#endif
```

4.2 RTOS ???? (freertos.c)

```
/* freertos.c - 核心控制邏輯 */
#include "main.h"
#include "cmsis_os.h"
#include "can.h"

// 全域變數與 Handles
extern VehicleState_t g_VehicleState;
extern osMutexId_t VehicleMutexHandle;
extern CAN_HandleTypeDef hcan1;

// 輔助函式: 發送 CAN
void VESC_Send_Current(uint8_t controller_id, int32_t current_ma) {
    CAN_TxHeaderTypeDef TxHeader;
    uint32_t TxMailbox;
    uint8_t TxData[4];

    TxHeader.ExtId = (CAN_PACKET_SET_CURRENT << 8) | controller_id;
    TxHeader.IDE = CAN_ID_EXT;
    TxHeader.RTR = CAN_RTR_DATA;
    TxHeader.DLC = 4;

    // Big Endian Packing
    TxData[0] = (uint8_t)(current_ma >> 24);
    TxData[1] = (uint8_t)(current_ma >> 16);
    TxData[2] = (uint8_t)(current_ma >> 8);
    TxData[3] = (uint8_t)(current_ma);

    if (HAL_CAN_GetTxMailboxesFreeLevel(&hcan1) > 0) {
        HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox);
    }
}

// --- Task 1: 核心馬達控制 (1kHz) ---
void StartControlTask(void *argument)
{
    uint32_t tick_count = osKernelGetTickCount();
    const uint32_t period = 1; // 1ms

    // 本地變數 (減少 Mutex 佔用)
```

```
int32_t cmd_FL=0, cmd_FR=0, cmd_RL=0, cmd_RR=0;
int32_t loc_rpm=0, loc_thr=0, loc_steer=0;
uint8_t loc_mode=0;

for(;;)
{
    // 1. 快速讀取狀態
    if (osMutexAcquire(VehicleMutexHandle, 2) == osOK) {
        loc_rpm   = g_VehicleState.avg_erpm;
        loc_thr   = g_VehicleState.throttle_adc;
        loc_steer = g_VehicleState.steering_val;
        loc_mode  = g_VehicleState.drive_mode;
        osMutexRelease(VehicleMutexHandle);
    }

    // 2. 計算總功率預算 (Base Torque)
    // 使用位元移位 (>>12) 代替除以 4096, 效率極高
    int32_t total_ma = (loc_thr * MAX_TOTAL_CURRENT_MA) >> 12;

    // 3. 判斷驅動模式 (Hysteresis)
    if (loc_mode == 0) { // 4WD
        if (loc_rpm > (ERPM_THRESHOLD_RWD + ERPM_HYSTERESIS)) loc_mode = 1;
    } else { // RWD
        if (loc_rpm < (ERPM_THRESHOLD_RWD - ERPM_HYSTERESIS)) loc_mode = 0;
    }

    // 4. 計算前後軸基底電流
    int32_t base_front = 0;
    int32_t base_rear  = 0;

    if (loc_mode == 0) { // 4WD Mode
        // 策略: 前 40%, 後 60%
        base_front = (total_ma * 40) / 100;
        base_rear  = total_ma - base_front;
    } else { // RWD Mode
        base_front = 0; // 前輪空轉
        base_rear  = total_ma; // 後輪全功率
    }
```

```
// 5. 計算電子差速與向量修正 (Differential & Vectoring)
// 簡單模型: 轉向值 * 係數 = 電流差值
int32_t diff_adj = (loc_steer * STEERING_GAIN) / 100;

// 6. 混合與分配 (Mixing)
// 前軸 (注意: 離合器不接受負值)
int32_t fl_temp = (base_front / 2) + diff_adj;
int32_t fr_temp = (base_front / 2) - diff_adj;

// Clamp logic for front clutch (No regen)
cmd_FL = (fl_temp < 0) ? 0 : fl_temp;
cmd_FR = (fr_temp < 0) ? 0 : fr_temp;

// 後軸 (接受負值, 允許向量煞車)
// 這裡可以加入額外的 IMU 穩定修正 (Yaw Control)
cmd_RL = (base_rear / 2) + diff_adj;
cmd_RR = (base_rear / 2) - diff_adj;

// 7. 最終安全檢查 (Total Current Limiter)
// 防止向量控制疊加後超過電池極限
int32_t total_req = cmd_FL + cmd_FR + cmd_RL + cmd_RR;
if (total_req > MAX_TOTAL_CURRENT_MA) {
    // 簡單等比例縮小 (Scaling Down)
    // 這裡為了效率, 簡化處理:
    // 實際專案建議使用定點數乘法進行縮放
}

// 8. 回寫模式狀態
if (loc_mode != g_VehicleState.drive_mode) {
     if (osMutexAcquire(VehicleMutexHandle, 0) == osOK) {
        g_VehicleState.drive_mode = loc_mode;
        osMutexRelease(VehicleMutexHandle);
    }
}

// 9. 發送 CAN 指令
VESC_Send_Current(VESC_ID_FL, cmd_FL);
VESC_Send_Current(VESC_ID_FR, cmd_FR);
VESC_Send_Current(VESC_ID_RL, cmd_RL);
```

```
        VESC_Send_Current(VESC_ID_RR, cmd_RR);

        // 10. 精確等待
        tick_count += period;
        osDelayUntil(tick_count);
    }
}
```

5. ????????? (Wiring Notes)

1. CAN Bus終端電阻：確保CAN H/L兩端各有一顆120Ω電阻(MCU端與最遠端VESC)。
2. 共地(Common Ground): 所有的VESC訊號地(GND)必須與STM32的GND連接。
3. VESC設定(VESC Tool):
    - App Settings: 設為 `CAN Uart` 或 `CAN`。
    - Controller ID: 必須分別設為1, 2, 3, 4且不重複。
    - Baud Rate: 設為 `500K` 或符合STM32設定。
    - Current Limits: 電機最大電流(Motor Current Max)可設為20A (500W級距), 但輸入電流(Battery Current Max)需配合電池BMS限制。

6. ??????

1. 弱磁控制(Field Weakening): 在RWD模式下, 透過VESC Tool開啟後輪的弱磁功能, 利用集中的1kW功率突破基速限制。
2. 牽引力控制(TCS): 在Task_Control中加入輪速比較, 若(Wheel_RPM - Avg_RPM) > Threshold, 則瞬間將該輪電流cmd減半。
3. IMU融合濾波：使用互補濾波器融合加速度計與陀螺儀, 獲得更穩定的Yaw Rate, 提升高速過彎穩定性。

Revision #3
Created 2026-04-01 02:06:16 UTC by TaipeiTechRacing
Updated 2026-04-11 14:33:40 UTC by AI Agent