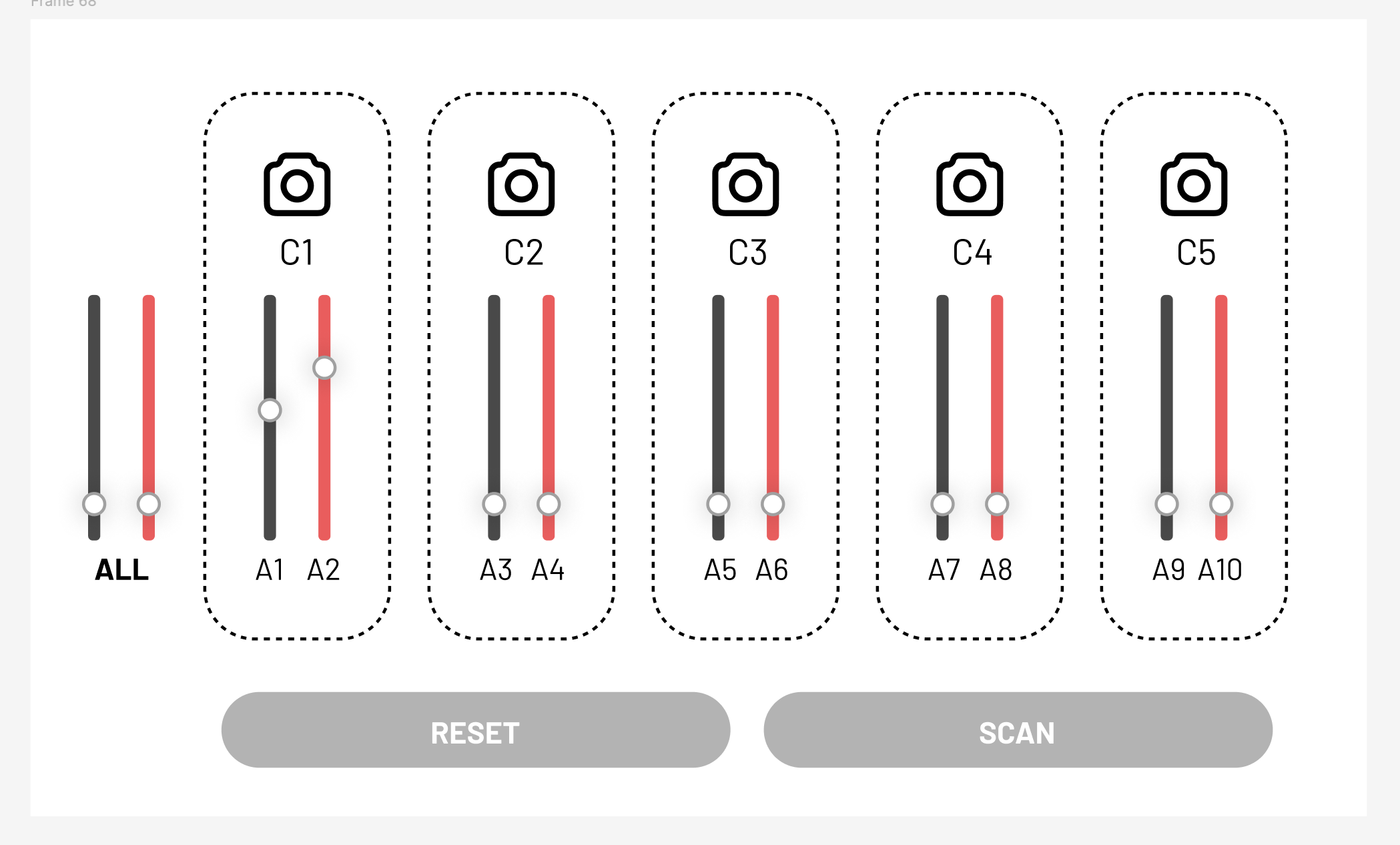
****

**图1 UI图**

**1.现有代码：**

**1.1目前的分组顺序：**

const int motorPins[10][3] = {

{4, 5, 6}, {7, 15, 16}, {17, 18, 8}, {19, 20, 3}, {46, 9, 10},

{11, 12, 13}, {14, 21, 47}, {48, 45, 35}, {38, 37, 36}, {41, 40, 39}};

**1.2目前的控制指令：**

{"state":0,"target":[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]}

**2.新的代码需求：**

**2.1功能需求**

**在同一时间内，控制五个电机的整体移动角度，控制五个摄像头的移动角度。**

**2.2代码需求：**

我们需要将**引脚划分为两组，每组包含五个电机**。一组电机为**整体移动角度**，另一组为**摄像头移动角度**。

如：const int motorPins1[5][3] = {

{4, 5, 6}, {7, 15, 16}, {17, 18, 8}, {19, 20, 3}, {46, 9, 10}};

const int motorPins2[5][3] = {

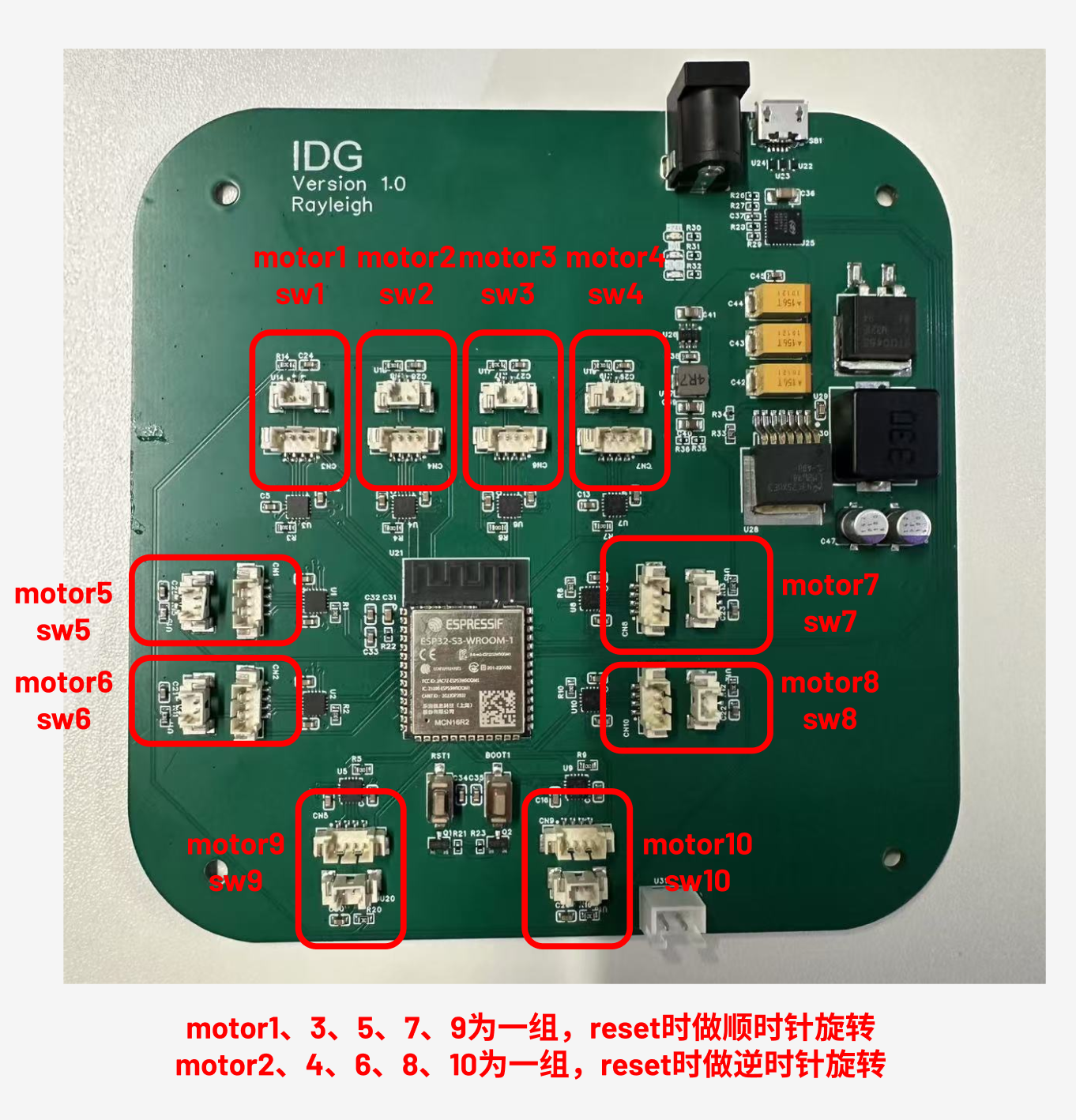
{11, 12, 13}, {14, 21, 47}, {48, 45, 35}, {38, 37, 36}, {41, 40, 39}};

我们需要将控制电机移动的代码修改为两部份，一部份为**整体移动角度**，另一部份为**摄像头移动角度**。

如：{"start","state":0,"target":[100, 200, 300, 400, 500]}

{"camera","state":0,"target":[100, 200, 300, 400, 500]}

另外，我们需要获取pcb上所有引脚与电机的对立关系，例如：



**3.控制电机**

**3.1 用户操作**

用户输入**角度1**，用于控制整体结构。角度的范围为**0-70°**。

如：{"start","state":0,"target":[0, 20, 35, 40, 70]}

随后输入**角度2**，用于控制摄像头。角度的范围为**0-120°**。

如：{"camera","state":0,"target":[120, 80, 60, 40, 80]}

用户点击回车键输入指令（点击**SCAN**），开始运行指令。

**3.2代码运行逻辑：**

1. reset：当用户输入控制指令后，motorPins1按**顺时针**方向持续运动，直至激活switch。motorPins2按**逆时针**方向持续运动，直至激活switch。此时我们完成了整体结构和摄像头的复位，让它们都回到了位于switch的原点。
2. 判定switch是否被触发：如果switch被触发，即执行下一步。
3. 移动到指定位置：根据用户输入的两个角度信息，控制电机运动。其中motorPins1按逆**时针**方向持续运动，直至到达指定位置。motorPins2按顺时针方向持续运动，直至到达指定位置。

**4.控制电机**

**4.1 用户操作**

用户输入指令，用于重置所有整体结构和摄像头的位置。

如：{"start","state":1,"target":[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]}

用户点击回车键输入指令（点击**RESET**），开始运行指令。

**4.2 代码运行逻辑**

reset：当用户点击reset按钮后，**无论电机处于什么状态，都使**motorPins1按**顺时针**方向持续运动，直至激活switch。motorPins2按**逆时针**方向持续运动，直至激活switch。完成整体结构和摄像头的复位，让它们都回到位于switch的原点。