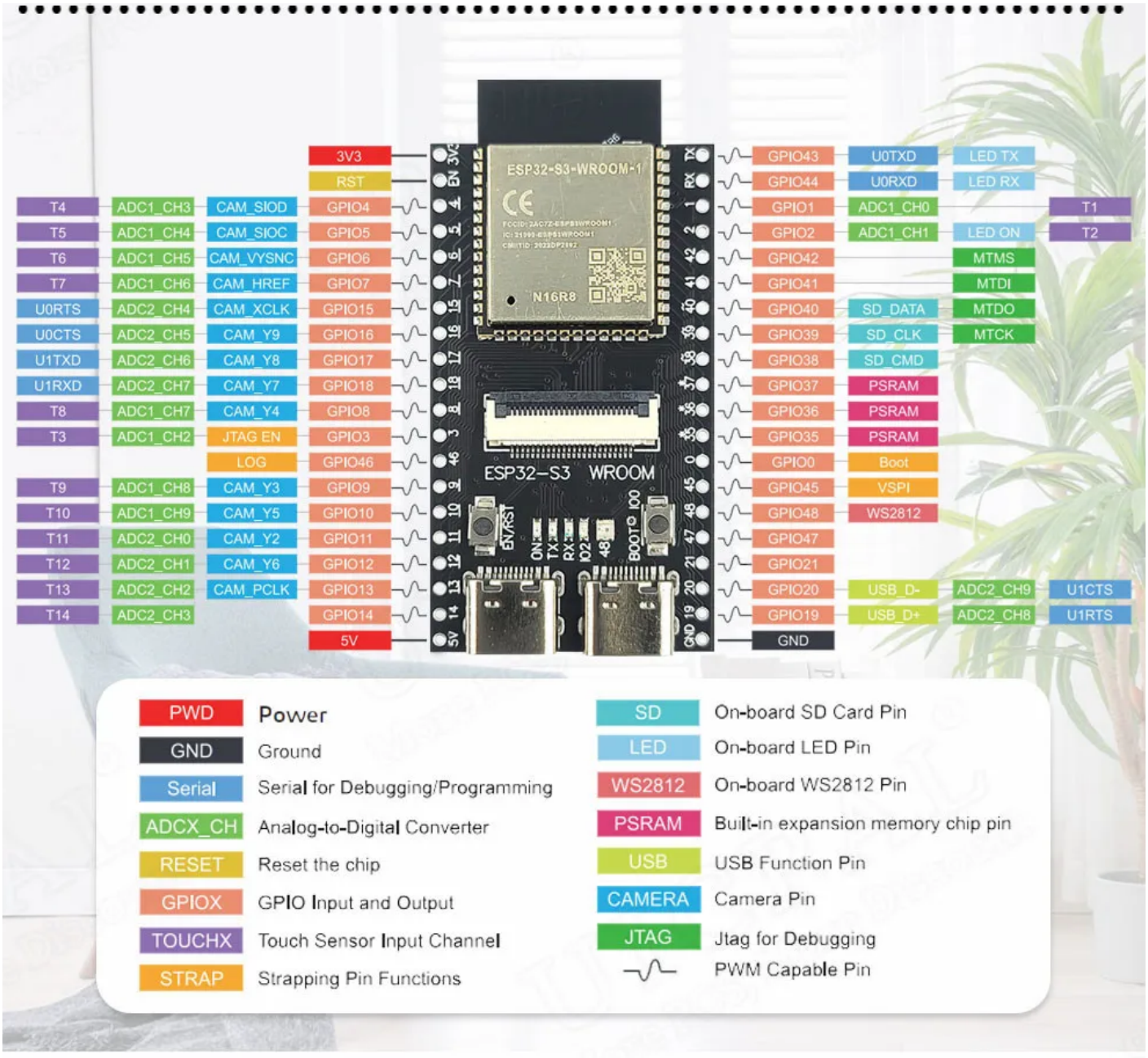
SD Card interfacing

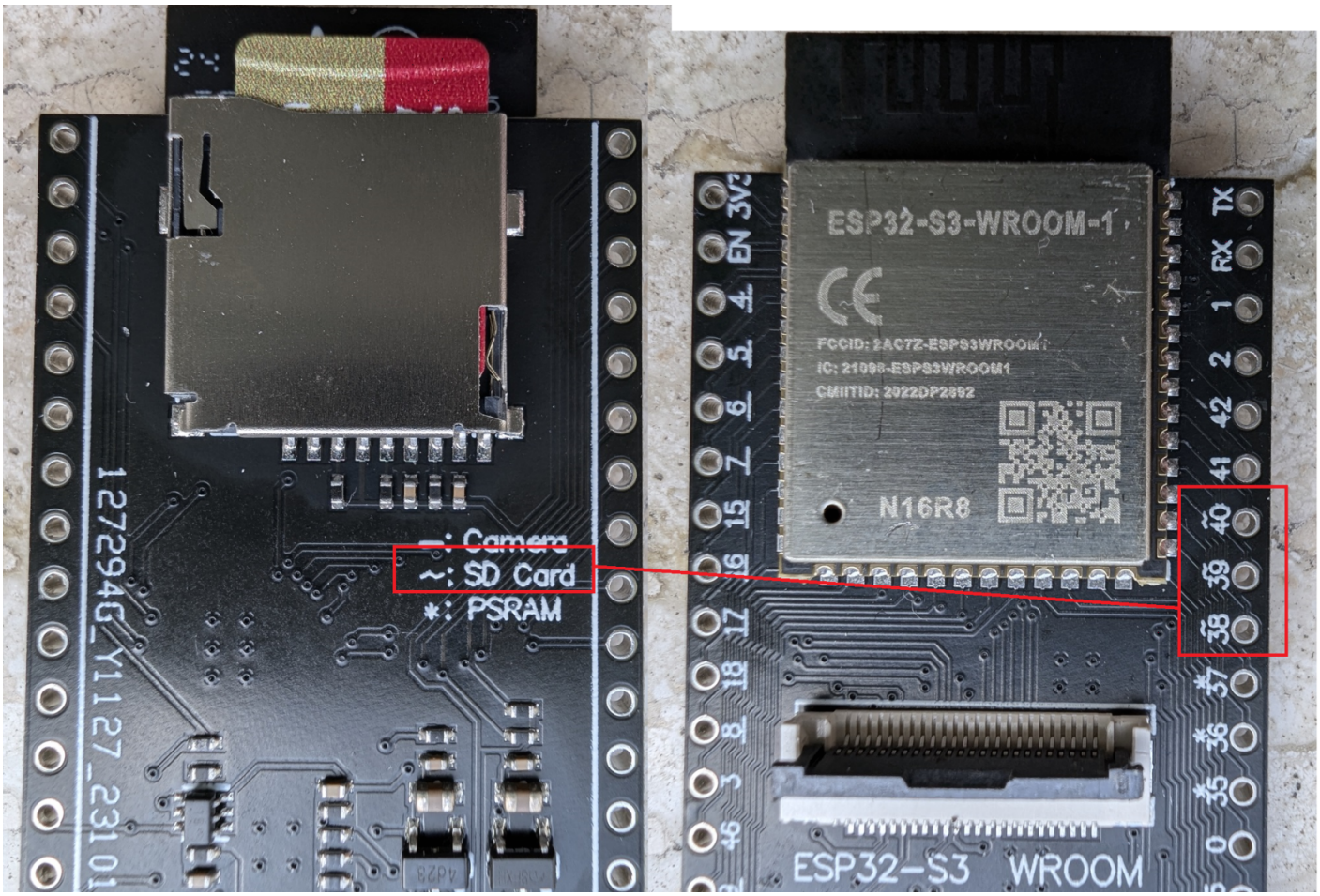
Trying to interface with this onboard SD card slot didn't go as easily as I had expected: it requires a different approach as the basic sample. It only works in "single bit mode".

The first thing to do, is to find the pins that are relevant.

I could find for this board this schematic: notice in the right blue blocks we have "SD_DATA", "SD_CLK" and "SD_CMD"



When we inspect the board itself, we can see a small legend that doesn't seem to make so much sense, until we flip the board around and can find the icons next to the port. So this gives a clue where to start.



We now know there are 3 pins that are used for the SD-card, namely 38, 39, 40.

We can now look for code samples that include these pinouts or write our own.

The below example shows working code to write text to the SDCard.

```
#include "esp_vfs_fat.h"
#include "driver/sdmmc_host.h"
#include "sdmmc_cmd.h"
#include "SD.h"

static const char *TAG = "example";
#define MOUNT_POINT "/sdcard"
#define EXAMPLE_MAX_CHAR_SIZE    64

esp_err_t ret;
```

```
void setup() {
  Serial.begin(115200);
  // Initialize SDMMC host driver
  sdmmc_host_t host = SDMMC_HOST_DEFAULT();
  sdmmc_slot_config_t slot_config = SDMMC_SLOT_CONFIG_DEFAULT();
  slot_config.width = 1;
  slot_config.clk = GPIO_NUM_39;
  slot_config.cmd = GPIO_NUM_38;
  slot_config.d0 = GPIO_NUM_40;

  esp_vfs_fat_sdmmc_mount_config_t mount_config = {
    .format_if_mount_failed = true,
    .max_files = 5,
    .allocation_unit_size = 16 * 1024
  };
  sdmmc_card_t* card;
  esp_err_t err = esp_vfs_fat_sdmmc_mount("/sdcard", &host, &slot_config, &mount_config, &card);

  if (err != ESP_OK) {
    Serial.printf("Failed to mount SD card (%s)\n", esp_err_to_name(err));
    return;
  }
  sdmmc_card_print_info(stdout, card);

  // First create a file.
    const char *file_hello =  MOUNT_POINT"/hello.txt";
    char data[EXAMPLE_MAX_CHAR_SIZE];
    snprintf(data, EXAMPLE_MAX_CHAR_SIZE, "%s %s!\n", "Hello", card->cid.name);
    ret = s_example_write_file(file_hello, data);
    if (ret != ESP_OK) {
        return;
    }
}
void loop()
{

}
static esp_err_t s_example_write_file(const char *path, char *data)
{
    ESP_LOGI(TAG, "Opening file %s", path);
    FILE *f = fopen(path, "w");
```

```
    if (f == NULL) {
        ESP_LOGE(TAG, "Failed to open file for writing");
        return ESP_FAIL;
    }
    fprintf(f, data);
    fclose(f);
    ESP_LOGI(TAG, "File written");

    return ESP_OK;
}

static esp_err_t s_example_read_file(const char *path)
{
    ESP_LOGI(TAG, "Reading file %s", path);
    FILE *f = fopen(path, "r");
    if (f == NULL) {
        ESP_LOGE(TAG, "Failed to open file for reading");
        return ESP_FAIL;
    }
    char line[EXAMPLE_MAX_CHAR_SIZE];
    fgets(line, sizeof(line), f);
    fclose(f);

    // strip newline
    char *pos = strchr(line, '\n');
    if (pos) {
        *pos = '\0';
    }
    ESP_LOGI(TAG, "Read from file: '%s'", line);

    return ESP_OK;
}
```

Revision #1
Created 16 September 2024 12:34:19 by Tim
Updated 25 September 2024 06:05:06 by Tim