Ecriture du firmware

Utilisation de la bibliothèque RMT pour envoyer un nombre d'impulsions PWM précis.

```
#include "M5Unified.h"
#include "Module_Stepmotor.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/rmt.h"

static Module_Stepmotor driver;

#define RMT_X_AXIS_CHANNEL RMT_CHANNEL_0    // Use RMT channel 0
#define RMT_Y_AXIS_CHANNEL RMT_CHANNEL_1    // Use RMT channel 1
#define RMT_Z_AXIS_CHANNEL RMT_CHANNEL_2    // Use RMT channel 2

#define PWM_FREQUENCY 15000             // Desired frequency in Hz
#define PULSE_COUNT 200                 // Exact number of pulses

void setup() {
  Serial1.begin(115200, SERIAL_8N1, 35, 5);
  Serial2.begin(115200, SERIAL_8N1, 34, 26);
  Serial2.setTimeout(100);

  M5.begin();
  Wire.begin(21, 22, 400000UL);
  driver.init(Wire);
  driver.resetMotor(0, 0);
  driver.resetMotor(1, 0);
  driver.resetMotor(2, 0);
  driver.enableMotor(1);

  // Motor DIR
  pinMode(17, OUTPUT);          // Axe X
  pinMode(13, OUTPUT);          // Axe Y = -X
  pinMode(0, OUTPUT);           // Axe Z
```

```
// Motor DIR
digitalWrite(17, 0);          // Axe X
digitalWrite(13, 1);          // Axe Y = -X
digitalWrite(0, 0);           // initialize in upward direction (Vz > 0)

// Convert frequency to time in microseconds
int period_us = 1000000 / PWM_FREQUENCY;  // Full period in microseconds
int high_time = period_us / 2;  // 50% duty cycle
int low_time = period_us / 2;

// Configure RMT X AXIS
rmt_config_t rmt_x_axis;
rmt_x_axis.channel = RMT_X_AXIS_CHANNEL;
rmt_x_axis.gpio_num = GPIO_NUM_16;
rmt_x_axis.mem_block_num = 1;  // 1 memory block
rmt_x_axis.clk_div = 80;  // 1 µs per tick (80 MHz / 80)
rmt_x_axis.tx_config.loop_en = false;  // Do not loop
rmt_x_axis.tx_config.carrier_en = false;
rmt_x_axis.tx_config.idle_output_en = true;
rmt_x_axis.tx_config.idle_level = RMT_IDLE_LEVEL_LOW;
rmt_x_axis.rmt_mode = RMT_MODE_TX;

// Apply configuration
rmt_config(&rmt_x_axis);
rmt_driver_install(RMT_X_AXIS_CHANNEL, 0, 0);

// Configure RMT Y AXIS
rmt_config_t rmt_y_axis;
rmt_y_axis.channel = RMT_Y_AXIS_CHANNEL;
rmt_y_axis.gpio_num = GPIO_NUM_12;
rmt_y_axis.mem_block_num = 1;  // 1 memory block
rmt_y_axis.clk_div = 80;  // 1 µs per tick (80 MHz / 80)
rmt_y_axis.tx_config.loop_en = false;  // Do not loop
rmt_y_axis.tx_config.carrier_en = false;
rmt_y_axis.tx_config.idle_output_en = true;
rmt_y_axis.tx_config.idle_level = RMT_IDLE_LEVEL_LOW;
rmt_y_axis.rmt_mode = RMT_MODE_TX;

// Apply configuration
rmt_config(&rmt_y_axis);
```

```
  rmt_driver_install(RMT_Y_AXIS_CHANNEL, 0, 0);

  // Configure RMT Z AXIS
  rmt_config_t rmt_z_axis;
  rmt_z_axis.channel = RMT_Z_AXIS_CHANNEL;
  rmt_z_axis.gpio_num = GPIO_NUM_15;
  rmt_z_axis.mem_block_num = 1;  // 1 memory block
  rmt_z_axis.clk_div = 80;  // 1 µs per tick (80 MHz / 80)
  rmt_z_axis.tx_config.loop_en = false;  // Do not loop
  rmt_z_axis.tx_config.carrier_en = false;
  rmt_z_axis.tx_config.idle_output_en = true;
  rmt_z_axis.tx_config.idle_level = RMT_IDLE_LEVEL_LOW;
  rmt_z_axis.rmt_mode = RMT_MODE_TX;

  // Apply configuration
  rmt_config(&rmt_z_axis);
  rmt_driver_install(RMT_Z_AXIS_CHANNEL, 0, 0);

  // Create RMT items (each pulse is 2 items: HIGH then LOW)
  rmt_item32_t pulse_wave[PULSE_COUNT];

  for (int i = 0; i < PULSE_COUNT; i++) {
      pulse_wave[i].duration0 = high_time;  // High for half period
      pulse_wave[i].level0 = 1;  // HIGH
      pulse_wave[i].duration1 = low_time;  // Low for half period
      pulse_wave[i].level1 = 0;  // LOW
  }

  // Send the pulses
  rmt_write_items(RMT_X_AXIS_CHANNEL, pulse_wave, PULSE_COUNT, false);
  rmt_write_items(RMT_Y_AXIS_CHANNEL, pulse_wave, PULSE_COUNT, false);
  rmt_write_items(RMT_Z_AXIS_CHANNEL, pulse_wave, PULSE_COUNT, false);
}

 void loop() {
  // Nothing needed in loop; the RMT handles everything
 }
```

Revision #1
Created 31 January 2025 13:54:22 by Stephane Muller

Updated 31 January 2025 13:56:39 by Stephane Muller