Motorisation banc de mesure

Il s'agit de motoriser un instrument de mesure fixé à un portique mobile au dessus d'une carotte géologique. C'est donc un mouvement 1,5 degrés de liberté : X (deux moteurs pas-à-pas à contrôler) + Y sur deux positions (haut-bas), plus un déclenchement de la mesure.

Mise en œuvre M5Stack + Driver Module

J'utilise un M5Stack BASIC v2.6. Il faut dans l'IDE Arduino de choisir un M5Stack-Station dans le type de microcontroleur, et limiter les vitesses d'upload.

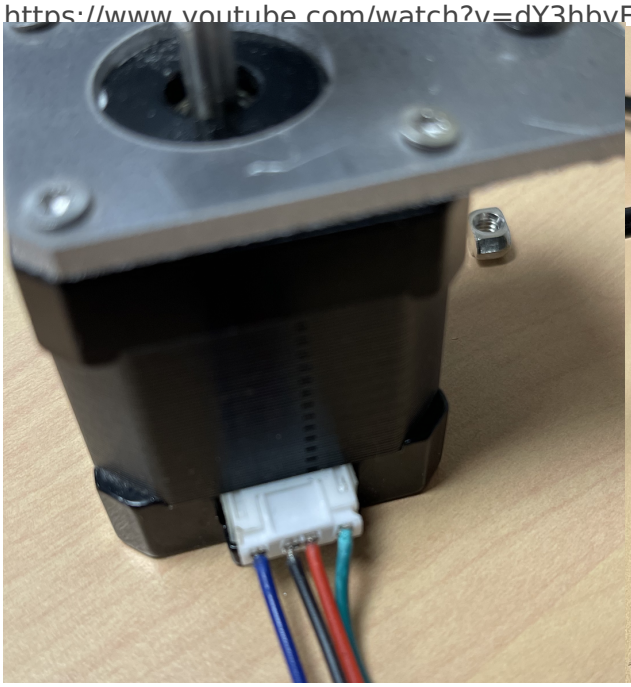
D'autre part, j'utilise le Stepmotor Driver Module 13.2. Il dispose de 3 drivers X,Y,Z et de 4 switchs de fin de course.

C'est donc en principe suffisant pour ce que je cherche à faire.

Les moteurs sont des MOTECH MT-170HS168A, à 1,8°/pas. Les moteurs disposent de connecteurs 6 pins JST-PH (espacement 2.0mm), mais sont quadrupolaires... Après avoir acheté un coffret de connecteurs à sertir, on commence donc par identifier les bobines à l'aide d'un multimètre.

Le meilleur tutoriel pour avoir le bon geste de sertissage et éviter d'y passer des heures de frustration :

https://www.youtube.com/watch?v=dY3hbyF3ypA



Un M5Stack et module driver brûlés par faute d'un déplacement intempestif, on arrive à faire fonctionner l'exemple produit ici

```
/*
*******************************************************************************
* Copyright (c) 2021 by M5Stack
*                 Equipped with M5Core sample source code
*                         配套 M5Core 示例源代码
* Visit for more
information: https://docs.m5stack.com/en/module/stepmotor_driver
* 获取更多资料请访问: https://docs.m5stack.com/zh_CN/module/stepmotor_driver
*
* Describe: Stepmotor With Pluse.
* Date: 2021/11/14
*******************************************************************************
  StepMotor Driver Module TEST Example,Directly drive the motor using pulse
signals. 步进电机模块测试示例,使用脉冲信号直接驱动电机
*/
#include "M5Stack.h"
#include "Module_Stepmotor.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

// #define VERSION_1_0

String inputString  = "";
bool stringComplete = false;

static Module_Stepmotor driver;

void setup() {
    M5.begin(true, false, true, false);
    M5.Lcd.clear(TFT_BLACK);
    M5.Lcd.setTextSize(2);
    M5.Lcd.setTextColor(TFT_GREEN);
    M5.Lcd.setTextDatum(MC_DATUM);
    M5.Lcd.drawString("DIRECT_STEPMOTOR", 160, 10, 2);
    M5.Lcd.setTextColor(TFT_YELLOW);
    M5.Lcd.drawString("LIMIT IO STATUS", 160, 90, 2);
    M5.Lcd.drawString("DIR", 70, 220, 2);
```

```
#ifndef VERSION_1_0
    M5.Lcd.drawString("FAULT IO STATUS", 160, 160, 2);
    M5.Lcd.drawString("RST ALL", 160, 220, 2);
    M5.Lcd.drawString("1/1", 260, 220, 2);
#else
    M5.Lcd.drawString("STEP", 260, 220, 2);
#endif

    Wire.begin(21, 22, 400000UL);
    driver.init(Wire);

#ifndef VERSION_1_0
    driver.resetMotor(0, 0);
    driver.resetMotor(1, 0);
    driver.resetMotor(2, 0);
#else
    driver.setMicrostepResolution(DIRECT_STEPMOTOR::kMicrosteps16);
#endif

    driver.enableMotor(1);
    Serial1.begin(115200, SERIAL_8N1, 35, 5);
    Serial2.begin(115200, SERIAL_8N1, 34, 26);
    Serial2.setTimeout(100);

    ledcSetup(0, 10000, 8);
    ledcAttachPin(16, 0);
    ledcAttachPin(12, 0);
    ledcAttachPin(15, 0);
    ledcWrite(0, 127);

    pinMode(17, OUTPUT);
    pinMode(13, OUTPUT);
    pinMode(0, OUTPUT);

    digitalWrite(17, 1);
    digitalWrite(13, 1);
    digitalWrite(0, 1);
}

void loop() {
```

```
    static uint8_t step_dir                             = 1;
    static uint8_t reset_mtr                            = 0;
    static Module_Stepmotor::MicrostepResolution_t micro_res = Module_Stepmotor::kMicrosteps16;
    Serial1.print("Y");
    delay(25);
    M5.update();
    if (Serial2.available()) {
        char inChar = (char)Serial2.read();
        if (inChar == 'Y') {
            M5.Lcd.fillRect(105, 180, 120, 20, TFT_GREEN);
        }
    } else {
        M5.Lcd.fillRect(105, 180, 120, 20, TFT_RED);
    }
    if (M5.BtnA.wasPressed()) {
        step_dir = 1 - step_dir;
        digitalWrite(17, step_dir);
        digitalWrite(13, step_dir);
        digitalWrite(0, step_dir);
    }

    if (M5.BtnB.wasPressed()) {
#ifndef VERSION_1_0
        driver.resetMotor(0, 0);
        driver.resetMotor(1, 0);
        driver.resetMotor(2, 0);
        reset_mtr = 0;
#endif
    }

    if (M5.BtnC.wasPressed()) {
#ifndef VERSION_1_0
        if (reset_mtr == 0) {
            driver.resetMotor(0, 1);
            driver.resetMotor(1, 1);
            driver.resetMotor(2, 1);
        }
        driver.resetMotor(int(reset_mtr / 2), (reset_mtr & 0x01));
        reset_mtr += 1;
        if (reset_mtr == 6) reset_mtr = 0;
```

```
#else
        if (micro_res == DIRECT_STEPMOTOR::kMicrosteps16) {
            micro_res = DIRECT_STEPMOTOR::kMicrosteps8;
        } else {
            micro_res = DIRECT_STEPMOTOR::kMicrosteps16;
        }
        driver.setMicrostepResolution(micro_res);
#endif
    }

    driver.getExtIOStatus();
    if (driver.ext_io_status[3]) {
        M5.Lcd.fillRect(70, 45, 20, 20, TFT_RED);
    } else {
        M5.Lcd.fillRect(70, 45, 20, 20, TFT_GREEN);
    }

    if (driver.ext_io_status[2]) {
        M5.Lcd.fillRect(120, 45, 20, 20, TFT_RED);
    } else {
        M5.Lcd.fillRect(120, 45, 20, 20, TFT_GREEN);
    }

    if (driver.ext_io_status[1]) {
        M5.Lcd.fillRect(170, 45, 20, 20, TFT_RED);
    } else {
        M5.Lcd.fillRect(170, 45, 20, 20, TFT_GREEN);
    }

    if (driver.ext_io_status[0]) {
        M5.Lcd.fillRect(220, 45, 20, 20, TFT_RED);
    } else {
        M5.Lcd.fillRect(220, 45, 20, 20, TFT_GREEN);
    }
#ifndef VERSION_1_0
    driver.getFaultStatus();
    if (driver.fault_io_status[2]) {
        M5.Lcd.fillRect(95, 115, 20, 20, TFT_RED);
    } else {
        M5.Lcd.fillRect(95, 115, 20, 20, TFT_GREEN);
```

```
    }

    if (driver.fault_io_status[1]) {
        M5.Lcd.fillRect(145, 115, 20, 20, TFT_RED);
    } else {
        M5.Lcd.fillRect(145, 115, 20, 20, TFT_GREEN);
    }

    if (driver.fault_io_status[0]) {
        M5.Lcd.fillRect(195, 115, 20, 20, TFT_RED);
    } else {
        M5.Lcd.fillRect(195, 115, 20, 20, TFT_GREEN);
    }
#endif
}
```

Test des fins de course

Tout est dans le titre.

Les fins de courses comprennent trois pôles:

- C
- NO (normally open)
- NC (normally closed)

Le connecteur étant branché C -> GND et NC->L0, on observe bien l'effet de cette partie du code :

```
 driver.getExtIOStatus();
if (driver.ext_io_status[3]) {
    M5.Lcd.fillRect(70, 45, 20, 20, TFT_RED);
} else {
    M5.Lcd.fillRect(70, 45, 20, 20, TFT_GREEN);
}
```

Le rectangle concerné devient rouge quand le bouton est enfoncé.

Tentative de première programmation

Il m'apparaît que le module StepMotor Driver 13.2 dont je dispose est la version 1.0.

Le microcontrolleur pilotant les drivers est en effet un TCA9554 (et non un STM32 comme dans les versions ultérieures).

Il faut donc décommenter la directive

```
#define VERSION_1_0
```

Et là, patatras, plus rien ne compile.

Il semblerait que j'ai mis le bazar dans mon installation en mettant à jour les bibliothèques : le problème est probablement lié à la bibliothèque FastLED dont dépend la bibliothèque M5Display dont dépend la bibliothèque M5stack...

La "nouvelle" version de l'API ignore par exemple ledcAttachPin (deprecated) et utilise ledcAttach, avec des arguments différents. Sans parler de ledcSetup qui a changé. Bref, c'est la m... s'il faut ré-écrire l'exemple.