Software Engineer - On-site assignment

Welcome at the Monumental office! We’re very excited to have you here for this trial for the software engineer role.

Assignment

Over the next two days, your aim is to draw a square (and maybe more!) using our pen-plotter hardware.

Below you will find some firmware code to get started with the hardware. The code contains some setup (no need to understand it in-depth), and some example code in the `loop()` function to move the actuators. Let's get that running first and then work from there.

Next, you'll need to modify it such that you can control the motors using commands sent from your computer over serial (e.g. from a Python script). The idea is that the firmware does the low-level commands, while your application code (e.g. Python script) reasons about higher-level concepts such as drawing lines.

Now your aim is to draw a rectangle using the system. Can you parametrise the rectangle?

Extension

Once you’ve got this working, consider extending your system with some interesting feature:

- **Vision candidates**: it would be cool to add a vision component to the system. For instance, draw something yourself on a piece of paper, then take a picture of it with a camera, process it into lines and then make the pen-plotter draw the same thing.
- **Product/UX candidates**: what would it look like to have a UI for the system? How would you make it possible for the user to design and draw more

interesting shapes?

Setup

Below you will find some sample firmware code that gives basic control over the pen-plotter's actuators. You will probably want to use this as your starting point for controlling them.

Hardware

- The stepper motor has 200 steps, which the motor driver divides further down to 256 microsteps per full step.
- At the output of the motor, there is also a 5:1 gearbox, ergo a full rotation would be $200 * 256 * 5$ microsteps.
- As an example, if `stepperDriver.XACTUAL()` currently returned 0, then setting `stepperDriver.XTARGET(200 * 256 * 5)` would cause the output to move one complete rotation.

The linear actuator has an potentiometer which returns a value indicating how extended the actuator is. See the `readPosition()` function below.

Tips

You will probably want to do some calibration of the actuator positions.

The microcontroller can work with USB-C alone. In order to drive the actuators you will need to turn on the power supply (ask one of us for help if you need it!)

If you turn off the power supply after running the setup code (see `setup()` below), you will need to run setup again as the motor driver needs to be reinitialised (`setupMotor()`)

Arduino

You'll need to install Arduino — I recommended 1.8.19 and not the newer 2.xx versions, along with the Arduino-Pico core:

https://github.com/earlephilhower/arduino-pico?tab=readme-ov-file#installation

Arduino libraries

Libraries (These should all be available from the internal libraries manager in the IDE — no need to manually install)

- TMCStepper https://github.com/teemuatlut/TMCStepper
- ADS1X15 https://www.arduino.cc/reference/en/libraries/ads1×15
- Adafruit PWM Servo https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library

Sample firmware code

```
#include <TMCStepper.h>

#include <Wire.h>
#include "ADS1X15.h"
#include <Adafruit_PWMServoDriver.h>

constexpr pin_size_t pin_chipSelA = 21; // Mot0
constexpr pin_size_t pin_chipSelB = 18; // Enc0
constexpr pin_size_t pin_chipSelC = 25; // Mot1
constexpr pin_size_t pin_chipSelD = 24; // Enc1

constexpr pin_size_t pin_SPI0SCK = 22;
constexpr pin_size_t pin_SPI0MOSI = 19;
constexpr pin_size_t pin_SPI0MISO = 20;

constexpr pin_size_t in1Pin = 1;
constexpr pin_size_t in2Pin = 0;

// Stepper Motor OConfig
constexpr uint16_t MOTOR_CURRENT = 4000;
constexpr uint8_t DEFAULT_IRUN = 31;
constexpr uint8_t DEFAULT_IHOLD = 16;
constexpr uint8_t TMC_TOFFRUN = 4;
```

```
#define SPEED_RPS 1.0
#define RATIO_ACCEL 5.0
#define RAIIO_DECEL 5.0

constexpr float FULL_STEPS_PER_REV = 200.0;
constexpr float MICROSTEP_FACTOR = 256.0;

constexpr float DEFAULT_VEL = (FULL_STEPS_PER_REV * MICROSTEP_F
constexpr float DEFAULT_ACCEL = (FULL_STEPS_PER_REV * MICROSTEP_
constexpr float DEFAULT_DECEL = (FULL_STEPS_PER_REV * MICROSTEP_

constexpr float RS = 0.05;

TMC5160Stepper stepperDriver(pin_chipSelA, RS);

Adafruit_PWMServoDriver linearDriver(PCA9685_I2C_ADDRESS, Wire1
ADS1015 ADS(0x48, &Wire1);

void setupMotor() {
  stepperDriver.setSPISpeed(1000000);
  stepperDriver.reset();
  stepperDriver.toff(0);
  stepperDriver.rms_current(MOTOR_CURRENT);
  stepperDriver.ihold(DEFAULT_IHOLD);
  stepperDriver.irun(DEFAULT_IRUN);
  stepperDriver.en_pwm_mode(false);
  stepperDriver.VSTOP(10);
  stepperDriver.RAMPMODE(0);
  stepperDriver.TZEROWAIT(0);

  stepperDriver.shaft(0);
  stepperDriver.en_softstop(0);
  stepperDriver.shortdelay(true);
  stepperDriver.shortfilter(2);
  //
  // Sets the internal motion profile —- see datasheet
  stepperDriver.v1(0); // Use Trapezoid Only, disable first ram
  stepperDriver.a1(DEFAULT_ACCEL);
  stepperDriver.d1(DEFAULT_DECEL);
  stepperDriver.AMAX(DEFAULT_VEL);
  stepperDriver.VMAX(DEFAULT_ACCEL);
  stepperDriver.DMAX(DEFAULT_DECEL);

  stepperDriver.toff(TMC_TOFFRUN);
}

void setupI2C() {
  Wire1.setSDA(2);
  Wire1.setSCL(3);
  Wire1.begin();

  Serial.println("Wire 1 Begin");
  if (!ADS.begin())
    Serial.println("ADS Error");

  if (!linearDriver.begin())
    Serial.println("PWM Error");

  linearDriver.setPin(in1Pin, 0);
  linearDriver.setPin(in2Pin, 0);

  Serial.println("I2C Done");
}

void setup() {
  pinMode(pin_chipSelA, OUTPUT);
  SPI.setTX(pin_SPI0MOSI);
  SPI.setRX(pin_SPI0MISO);
  SPI.setSCK(pin_SPI0SCK);

  SPI.begin();
```

}

```
  Serial.begin(9600);
  while (!Serial); // Wait for USB monitor to open
  Serial.println("Online");

  setupMotor();
  setupI2C();

  if (stepperDriver.test_connection() != 0)
  {
    Serial.println("Driver not connected");
    while (1);
  }
}


int readPosition() {
  return ADS.readADC(1);
}

void linearRetract(uint16_t speed = 4095)
{
  linearDriver.setPin(in1Pin, speed);
  linearDriver.setPin(in2Pin, 0);
}

void linearExtend(uint16_t speed = 4095)
{

  linearDriver.setPin(in1Pin, 0);
  linearDriver.setPin(in2Pin, speed);
}

void linearStop() {
  linearDriver.setPin(in1Pin, 0);
  linearDriver.setPin(in2Pin, 0);
}
```

```
// Example of using the functions to control things
void loop() {
  Serial.println(readPosition()); // Read "FL" Port
  linearRetract();
  delay(1000);
  linearExtend();
  delay(1000);
  linearStop();
  delay(1000);

  // Move 10 full steps from current position
  stepperDriver.XTARGET(stepperDriver.XACTUAL() + (MICROSTEP_FAC
  // Wait until done
  while (!stepperDriver.position_reached()) {
    Serial.println("Moving");
    delay(1000);
  }
}
```