Device for controlling a stepper motor with a Raspberry Pi

The goal is a SoftMotion device that allows stepper motors to be controlled by position and speed using a Raspberry Pi and as little additional hardware as necessary.

Development

In CODESYS it is possible to control the general purpose input/output pins (GPIO pins) via a GPIO driver. These pins are read before each program run and written at the end of the run. This significantly delays the reaction to a change, which is why this implementation is not suitable for many applications, especially in the area of real-time control.

To enable such an application, the GPIO driver can be avoided by initializing the pins already in the program, setting them as input/output and controlling them. However, this is time-consuming, especially if several motors are to be controlled.

Therefore the control of the GPIO pins and the motor was outsourced to a library and converted to a SoftMotion device. It is now only necessary to add a SoftMotion device to the project for each stepper motor, enter the pins and the motor-specific parameters.

Commissioning

In this chapter the setup, the commissioning and the required hardware components in general are described. A concrete example can be found under "Function example".

Hardware types

Raspberry Pi:

A Raspberry Pi serves as a programmable logic controller (PLC). It is a single-board computer with the Linux-based operating system Raspbian. The Raspi can be extended with a real-time patch, which allows you to reserve processor cores for a specific application. As a result, deviations in e.g. the cycle time are significantly lower.

Motor driver:

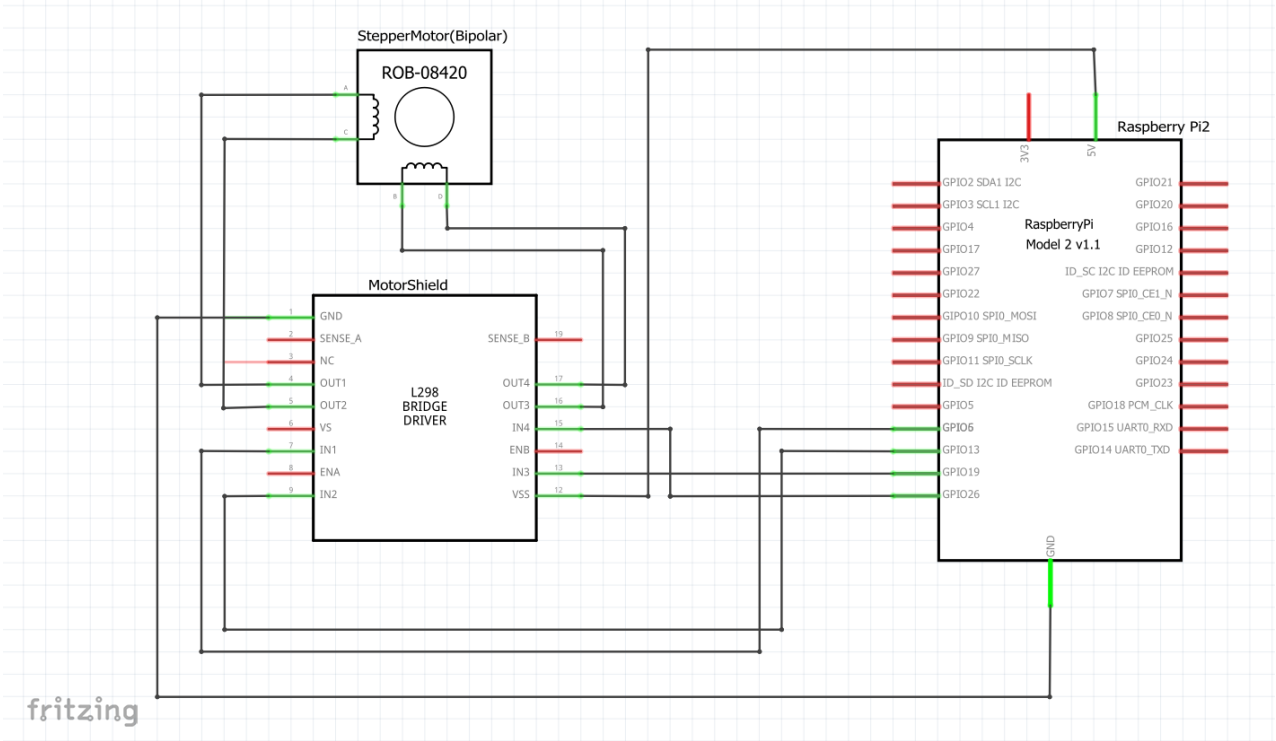
A motor driver is a circuit of transistors that amplifies the control current of the PLC as motor current. In addition, the motor driver protects the controller from short-circuit and induction currents of the motor and enables the motor to be supplied with more than 5V. The speed and direction can be controlled via a motor driver. In the example below, an L298N, a DRV8835 and an ULN2003 are used and compared, but there are also numerous other motor drivers with different electrical specifications that can be used to control stepper motors of different power.

Stepper motor:

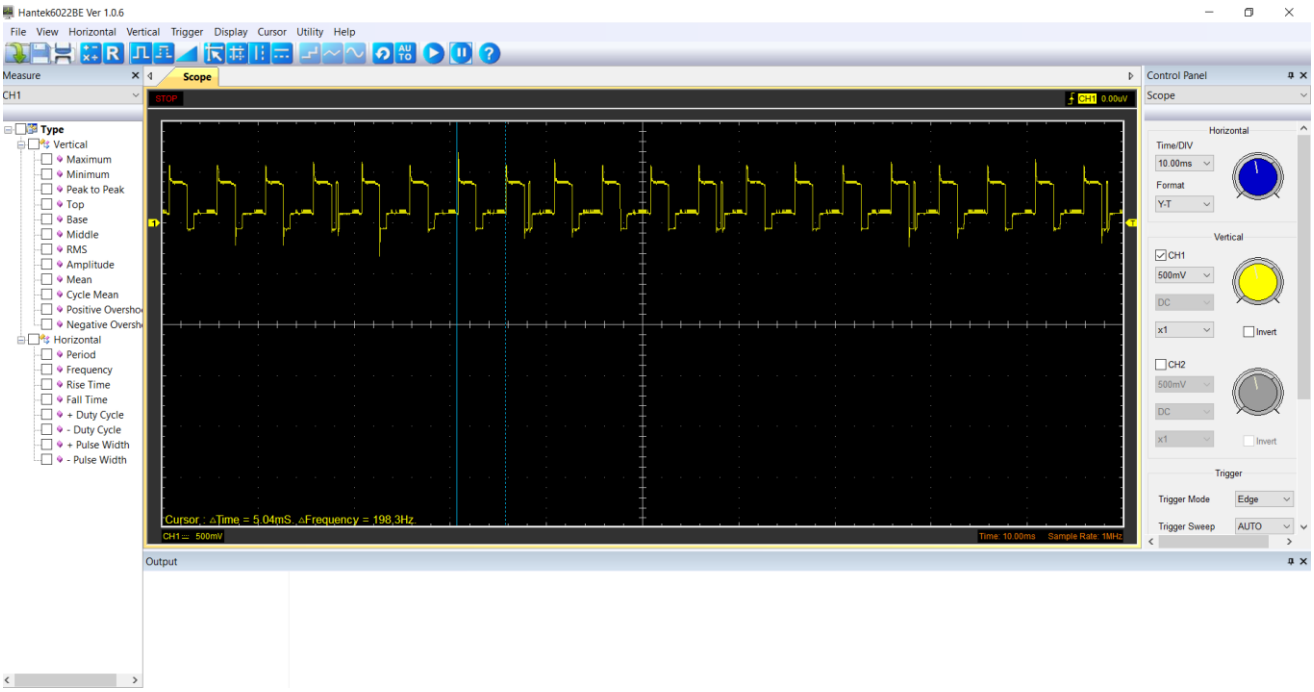
A stepper motor is a motor in which rotation is generated by a stepwise rotating electromagnetic field. In a bipolar stepper motor, each end of a coil is controlled by an output pin (A/A-, B/B-). In a unipolar stepper motor, one end of each coil is controlled by one pin, the other end is connected to its own potential. By controlling the motors in steps and half steps, it is possible to move them precisely to a desired position. The program runs through an array of different bytes, which contain a fixed logic pattern to control the individual motor pins.

Structure

The motor driver is connected to the GPIO pins of the controller. The motor and the supply voltage are applied to the motor driver as shown in the circuit diagram.



It is recommended to get the supply voltage for the stepper motor from an external source, because the 5V output of the Raspis is not constant enough. The Raspberry Pi is able to supply one or more motors, but even with only one motor you can see that there are small voltage drops and peaks. In both cases, a complete run of the logic array, i.e. 8 cycles, is marked in blue. The motor is operated in half steps and the "MotorPowerTime" is 1.



Supplied with the Raspberry Pi

Supplied with an external source

If you compare the signal from the PLC to the motor driver and from the motor driver to the motor, you will see that there is no time difference in the amplification of the signal. A cycle is marked in blue, the motor is operated at a frequency of 800Hz in half steps, which corresponds to a speed of 2 revolutions/seconds. The "MotorPowerTime" is 0.5.

Implementation

To get access to the SoftMotion device, the device description (devdes) has to be installed in the Device Repository in CODESYS. In the project "SM_Drive_RaspiStepper" must now be added for each individually controlled motor. These is done like all other SoftMotion axes. You have to click with the right mouse button on "SoftMotion General Axis Pool" and on "Add Device". There you will find the "SM_Drive_RaspiStepper" under "Softmotion drives", "position controled drives", which you add by clicking on "Add Device".

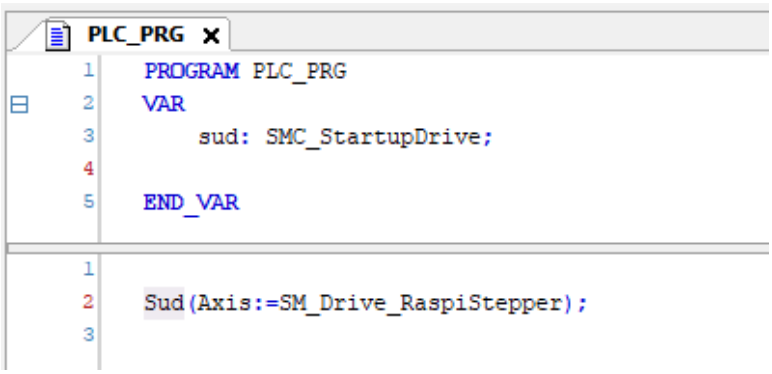
Double-click on the "SM_Drive_RaspiStepper" in the device tree to open the editor for the device. As with comparable SoftMotion devices, the motor-specific parameters such as the number of motor steps and the gear and motor ratio, maximum and minimum speed and acceleration must be set here. For the "SM_Drive_RaspiStepper" the GPIO pins to which the motor is connected via the motor driver must also be specified. The pins are inserted in the parameter list (SM_Drive_PosControl: Parameters). In addition, the "MotorPowerTime" must be specified. It determines per cycle which part of the cycle time the motor pins are supplied with current.

AXIS_REF: Motorsettings					
A	DWORD	6	6		MotorPin A
A-	DWORD	13	13		MotorPin A-
B	DWORD	19	19		MotorPin B
B-	DWORD	26	26		MotorPin B-
MotorPowerTime	LREAL	0.5	0.5		fraction of the cycle, where outputs are set High (determines the power of the motor)

You can also decide whether the motor should run in full or half steps. Operation in half steps is more precise, but requires twice as high a frequency as operation in full steps, since more steps must be taken for one revolution. Half step operation can be set in the SoftMotion device under Parameters, "Scalings" and "HalfStep", the position and frequency are automatically adjusted.

HalfStep	BOOL	TRUE	FALSE		Motor takes half steps instead of full steps

To call up the motor in the project, a program (e.g. PLC_PRG) must be added to the application as a "POU". There you create an instance of "SMC_StartupDrive" and assign this instance to the desired SoftMotion axis.



As the SoftMotion device and the previously mentioned GPIO driver both write in the GPIO library, it is not possible to use both of them in the same project.

Functionality

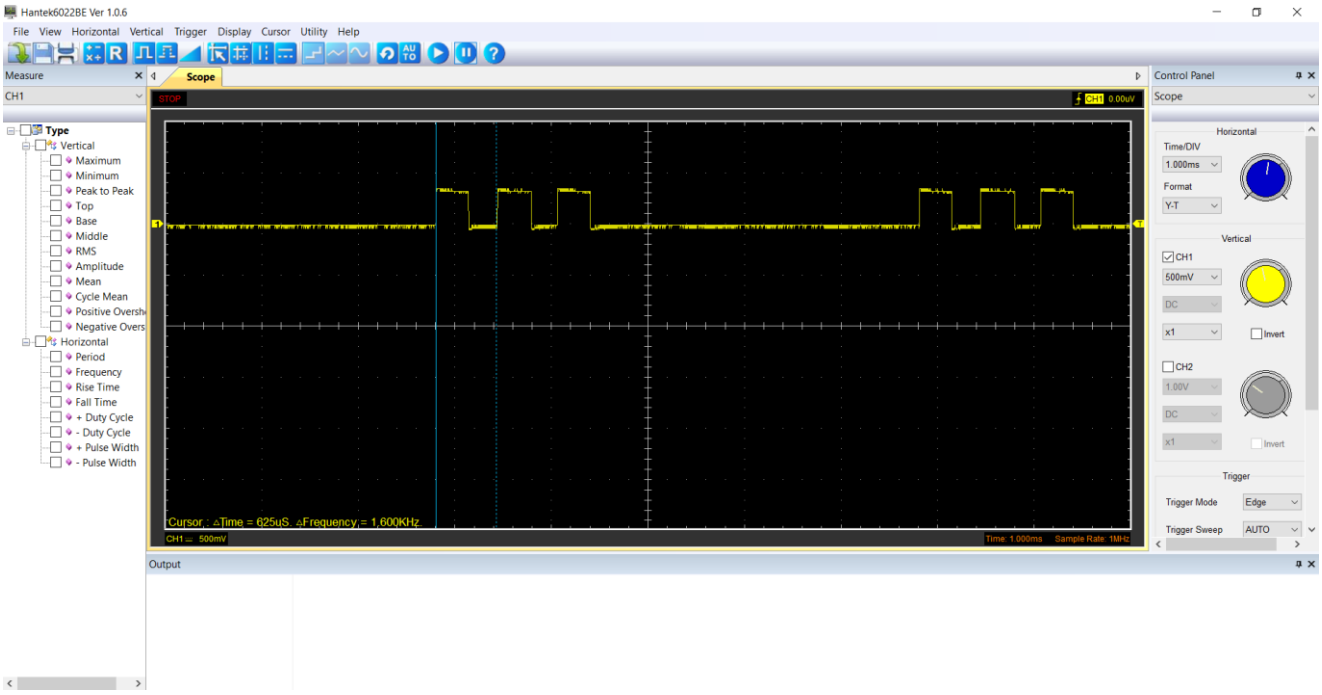
A separate task is created in each SoftMotion axis, which runs in the background and calls the StepperControl function block (FB). In this FB the set GPIO pins are first declared as outputs. Then the cycle time of the task is defined via the inverse value of a frequency. The frequency is determined by

the set speed and the ratio of the number of motor steps to the desired unit in the application. The cycle time is converted from seconds to microseconds.

Each cycle run is counted and added or subtracted depending on the direction of rotation. For the next step, in half step mode the modulo is taken from the counter and 8, in full step mode the modulo is calculated from the counter and 4 and then multiplied by 2 to obtain only every second value. The result of this determines the next byte from the logic grid and sends it to the GPIOs.

```
25        OutMask: ARRAY[0..7] OF BYTE := [2#1000, 2#1010, 2#0010, 2#0110, 2#0100, 2#0101, 2#0001, 2#1001];
```

The logic pattern is also visible on each Raspberry Pi output. In this measurement, the "MotorPowerTime" is 0.5, so the output is low 50% of the cycle time, so the individual cycles can be seen. One cycle is marked blue here, the motor runs in half step and has a frequency of 1.6 kHz, which corresponds to a speed of 4turns/second.



The position is determined from the number of cycles passed and the ratio of the number of motor steps to motor turns. In full-step operation, the number of cycles is calculated directly with the ratio; in half-step operation, the number of cycles is halved before it is calculated because twice as many half steps are required for one turn as full steps and therefore more cycles must be run through.

The windings of the individual coils are not energized for the whole cycle to preserve them. From the cycle time of one step, a percentage is calculated which the motor is controlled by. How large the percentage share is, is determined by the parameter "MotorPowerTime". The remaining time no current flows through the coils. It is also possible to supply the motor with current for the entire cycle. To do this, set "MotorPowerTime" to 1 in the SoftMotion device parameters.

Functional example

First the library, the device description (devdesc) and the example project has to be checked out from cforge. Then the “SM3_Drive_PosControl.devdesc.xml” file from “devdesk has to be installed in CODESYS in “Tools”, “Device Repository” to get access to the SoftMotion device. The project contains the control of two motors and a SoftMotion visualization for both, which can be used to call up the various SoftMotion FBs to turn the motors. A Raspberry Pi 2 Model B and a Raspberry Pi 4 with real-time patch is used for comparison. The motor used is a QSH2818-32-07-006 stepper motor, which is operated with an L298N and a DRV8835 for testing purposes, as well as a 28BYJ-48 stepper motor, which is operated with an ULN 2003.

The hardware is always connected according to the same principle described above.

Used hardware

The motor QSH2818-32-07-006 is a bipolar stepper motor with a step angle of 1.8°, i.e. it takes 200 steps for one turn. The motor has a rated voltage of 3.8V and a rated phase current of 0.67A.

The 28BYJ-48 is available from various suppliers and therefore has no uniform specifications for power supply and transmission between motor and axis, despite identical serial number. Here you should go according to the manufacturer's specifications, or measure it yourself.

The L298N needs a supply voltage of 5V. You can run a motor with a voltage of 5-35V, a current of 2A and a maximum power of 25W.

With the DRV8835 from Pololu you can control a stepper motor with a motor voltage up to 11V and a current of 1.2A on each line. It requires a logic voltage of 2-7V.

The ULN2003 is often used with the 28BYJ-48. It can be used to drive a motor with a voltage up to 50V and a current up to 500mA on each line.

Hardware comparison

The following tables compare the minimum and maximum frequency and speed of the QSH2818-32-07-006. Vary the PLC, using a Raspi2 and a Raspi4, and the motor driver, switching between a L298N and a DRV8835. The motor as well as the motor driver is supplied with a voltage of 5V. The "MotorPowerTime" is set to one.

In the first table the motors run in full step.

Component combination	Minimal frequency [Hz]; velocity [turns/second]	Maximal frecuency [Hz]; velocity [turns/second]
Raspi2, L298N	200; 1	1000; 5
Raspi2, DRV8835	600; 3	1200; 6
Raspi4, L298N	400; 2	1400, 7
Raspi4, DRV8835	200; 1	2400; 12

In the second table the motors run with half step, the other parameters are the same as in the first table.

Component combination	Minimal frequency [Hz]; velocity [turns/second]	Maximal frecuency [Hz]; velocity [turns/second]
Raspi2, L298N	200; 0.5	2400; 6
Raspi2, DRV8835	800; 2	2800; 7
Raspi4, L298N	400; 1	4400; 11
Raspi4, DRV8835	400; 1	7400; 19

With the two Raspberry Pis it is clear that the Raspi 4 has a better processor than the Raspi 2, when both motors are operated at maximum speed, the Raspi 4 is used to 6% and the Raspi 2 to 13%. The real time patch is also noticeable in the Pi4. The jitter (difference from the cycle time) is +-60µs for the Raspi4 with real-time patch and +-600µs for the Raspi2 without.