# Directions for use

Sometimes the motor needs a kick

Motor needs to be spinning to sing nicely

# Algorithms description

## Reading input from user

## Spinning at defined speed (V mode)

At the beginning, several initialisations are performed. The speed value read from user is converted to float variable. Based on a comparison with 0, flag for direction of spinning (clockwise or anticlockwise) is set and with is variable *lead*, polarity of which depends on the desired rotation direction. Next, the speed PID controller is set up with the correct constants for its P, I and D parts. Then, interrupts are enabled, speed timer is initialized and flag denoting mode od operation (speed control) is set. At the end speed control PID is started in a thread.

To spin the motor interrupt function *state\_interrupt* is triggered at each rising and falling edge of the photointerrupters, except the rising edge of the I1 photointerrupter (explained below). In the function, the rotor state is read from the photointerrupters and next state is output into the rotor fields. The state is set in another function, which finds the setting of the field transistors corresponding to the requested state, turns of all transistors (to avoid shoot-trough) and then sets them according to the state setting found before. The transistors are excited with PWM, calculated in the speed controller.

To calculate the PWM duty cycle PID controller from the mbed PID library is used. It is created, set up and started in a thread during initialisation of the program. The constants for the P, I and D parts were tuned during testing. The controller takes the input of the current speed (measurement of this is explained below) and the reference is the value requested by user. The control signal produced is saved into a distinct variable.

To measure the speed, interrupt function *state\_interrupt\_speed* is triggered on each rising edge of I1. It operates with a timer which is read and then restarted. Based on the timer output, current speed of the rotor is calculated and saved into appropriate variable. Afterwards, based on the mode-of-operation flags, appropriate duty cycle (in this case the one from speed controller) is assigned to a global variable. This is used in the function switching rotor field states as described above. At the end, next motor state is set as explained above.

Overall thus, the speed control mode consists of 2 parts. One is the PID controller running in a thread calculating appropriate PWM duty cycle to achieve desired speed. The other part is driven by interrupts from the photointerrupters. It spins the motor field around with application of the PWM pulse, and at each revolution calculates the speed of the rotor and updates the duty cycle of the PWM to be used.

From optimisation point of view, it was considered to set of the PWM duty cycle and calculate the speed in each of the interrupts (6 times a cycle). This way higher accuracy of the control could have potentially been achieved. However, this approach would also potentially pose problems at high speeds, where the new state might not get output to the transistors before the next interrupt is triggered. Moreover, the PWM calculation in the controller would have to be performed more often, thus taking more CPU time.

## Spinning for defined number of revolutions (R mode)

The mode for spinning for defined number of rotations is very similar in its operation to the V mode. It uses the same interrupts triggered by the photointerrupters and a PID controller calculating the required PWM duty cycle. At the beginning, desired number of revolutions from the user is read, and flags and variables for direction of spinning are set. Then interrupts are enabled, timer started, counter for number of revolutions initialised and flag for R mode set to 1. In the end position control PID controller is starter in a thread.

The position control PID is also an object created using the mbed PID library. The setpoint is the user-defined number of revolutions and measured variable number of revolutions already completed. The controller outputs PWM duty cycle, which is applied to the rotor fields in the same way as in V mode. The number of revolutions completed is tracked by a variable, which is incremented in the *state\_interrupt\_speed* function, i.e. once a revolution.

## Spinning for defined number of revolutions at defined speed (RV mode)

## Singing (T mode)

# High level program flow description

Threads

Deadlock analysis

CPU free time

# Tasks and threads analysis

## Execution time for threads

Measure latency from photointerrupt to motor

## Deadlines

### Photointerrupter to motor output

### PID output ready

## Thread priorities (scheduling)