Brushless DC engine controller

Project Report

Author: Michael Meißner

Introduction

Brushless DC motors are widely used in many low-cost applications, since they are a good compromise between the low maintenance of three-phase motors and the low control effort of DC engines. In high cost engine applications like industrial settings synchronous or asynchronous three-phase engines are used, because with modern field-oriented control, very high performance and dynamics with very low vibration can be archived. The downside to these is that field-oriented control needs a high computational effort in every control step, as well as extremely short cycle times, which results in very high hardware requirements for the controlling unit. On the other hand of the spectrum there are DC motors, where the DC current is commutated mechanically by brushes on the commutator plate. These are very simple to control, since one must only provide direct current to drive them. The downside is, that the live time of the brushes is very limited and they must be maintained often. Brushless DC motors (BLDC) are essentially permanent-magnet synchronous machines, that are optimized for being driven by square waves rather than sinus waves. They are essentially DC motors without a mechanical commutator, therefore power electronics must provide commutation.

This Project aims to develop a motor controller using a Microchip PIC16F1828 8-bit microcontroller as the core. The motor controller should provide sufficient commutation of the BLDC in a forced and a free mode as well as accept commands for the effective voltage the motor to drive the motor over a I2C interface. Additionally, it should monitor the current motor speed and respond to requests over the I2C interface with the current speed value. The controller behaves as the slave in the I2C interface. This project is roughly following [1], which describes the theory, but does not provide an implementation of a BLDC controller. Differences of this Project to [1] are a different hardware selection, as well as the added communication interface. Additionally, this Project uses an operational amplifier (op-amp) to provide information on the Back-EMF, rather than using the microchip's onboard comparator used in [1].

For validating the functionality, a second PIC microchip was used as the master, reading an analog signal from a potentiometer and sending commands accordingly to the motor controller. The master also reads the motor speed from the motor controller. Both is done periodically over the I2C interface.

All the work in this Project including hardware & software development were done by Michael Meißner.

BLDC Functionality

This paragraph is mainly based on [1].

BLDC motors are three-phase engines, optimized to function with square waves instead of sinus waves. Every electrical rotation of the engine (which must be executed the number of pole-pairs times to get one mechanical rotation) is divided into six sections. In every section one of the phases is driven high, one is driven low and the last one is floating. The current will then flow from the high phase to the low phase. These sections are then switched every 60 degrees, matching the speed of the motor. Since the effective voltage of the high level should be controlled by the motor controller, a plus-with-modulation

(PWM) between the high voltage level and ground is used when a phase is in the high-state instead of just putting the input voltage on the phase the whole time. The effective voltage of the high state is then determined by the plus with of the PWM. A diagram of the six sections can be seen in figure 1.

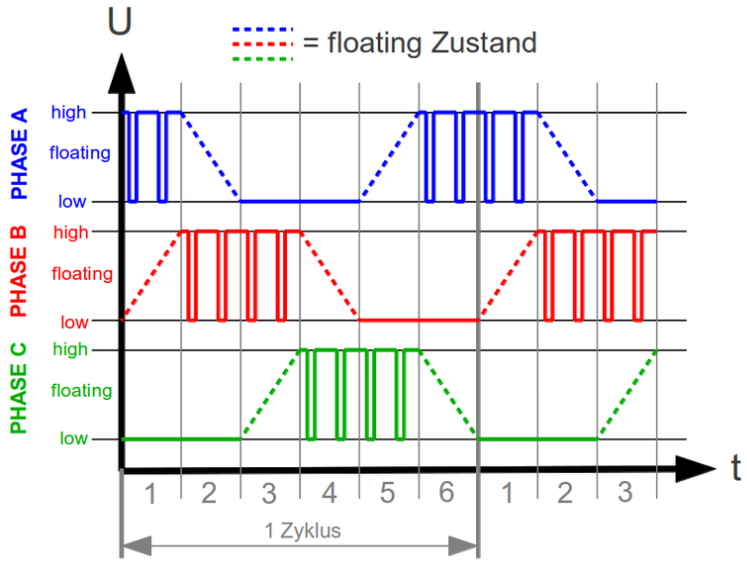

Figure 1: sections over one revolution; source: [1]

The difficulty here lies in knowing when to change the section, hence knowing at which position the motor is in its rotation. The switching of the sections can be done in two ways here referred as forced commutation and free commutation. In the forced commutation the sections are switched blindly in fixed intervals. This leads to high currents that result in very high idle power and power loss, since the engine gets accelerated and decelerated in every section. When free commutation is applied, the electrical angle of the motor is measured and it is determined when the correct time is to switch section according to the motor position. This is done automatically by hardware when a mechanical commutator is used in a DC motor. In a BLDC motor either a sensor (e.g. a hall sensor) can be used to determine the engine orientation, although in many low-cost applications such a sensor is not available. If no sensor is available, the back-electromagnetic-force (BackEMF) of the engine can be used to determine the time for the next commutation of the engine. In every section one phase is floating. When the engine is turning, it will induce a potential on the floating phase, ideally crossing the neutral-point-potential of the engine in the middle of the section. When the controller is comparing these two potentials it can determine the middle of the current section and therefore can time the next commutation, if it has measured the duration of each section. This method can be applied using no expensive sensor, but works only if the motor is already turning, since it otherwise induces no voltage.

The time of the commutation is critically for the operation of the motor. If it is too early the magnetic field in the motor is effectively advanced against its optimal position and is less effective in driving the rotor. This mode is called field weakening and can also be utilized, because in a less effective field the rotor will induce less BackEMF and therefore can turn faster but loses efficiency. The field weakening mode can also be used to simplify the commutation procedure, since it is possible to cumulate directly when a zero crossing of the BackEMF causes an interrupt without timing difficulties. If the commutation comes too late, the engine will be decelerated at the end of every section leading to high currents, high idle and dissipation powers much lick the forced commutation. If the commutation is much too late it has shown that the engine can come to a dead stop.

Hardware

The engine for that was choses for this project is the A2212/13T a 14-pol 150W motor developed for model aircraft and drones. It is typically used to drive small propellers.

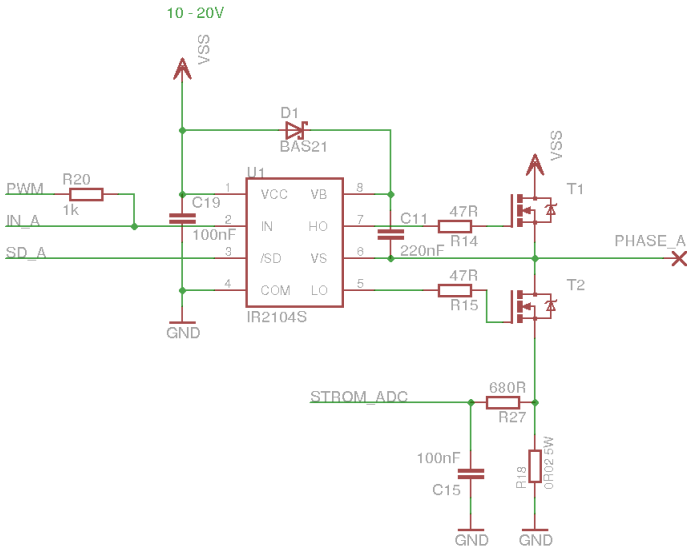

Figure 2: half-bridge plus driver; source: [1]

To drive the three phases of the motor, a half-bridge with a half-bridge-driver is used according to [1]. This is shown in figure 2. The current sensing unit (R27, R18, C15) was not implemented. The basic circuit-layout is taken from [1], but the selection of the components was done independently. The half-bridge-driver IR2109 was used together with two IFR510 CMOS transistors. The bootstrap circuit (D1 & C11) is used to create a switching-voltage for the high side transistor T1, since this voltage required is higher than VSS, the maximum voltage available in the circuit-board. When the phase is low, C11 is charged and when the phase is driven high the capacitor holds its charge and the has a potential of two times VSS on one side, which can then be used to switch T1. When SD is '1' the driver is active and is switching one of the transistors according to IN, otherwise both transistors are switched off, floating the phase. IN is pulled to the PWM pin of the microcontroller. When the phase is driven high the pin connected to IN_A in the tristate and therefore is pulled to the PWM signal. When the phase is driven low, the chip outputs a '0' on SD overriding the pull-up.

The diode 512-1N914TR for D1 was chosen for its low switching time. The transistors were chosen mainly for their high Drain-Source-Voltage (V_{GS}), as well as their high Drain-Source-Threshold-Voltage ($V_{GS(th)}$) of over 3V. In a previous iteration Transistors with a lower $V_{GS(th)}$ were used. This led to both transistors being switched on when the half-bridge-driver was switching between VSS and GND on the phase. This short-circuited GND and VSS and VSS fell under zero, killing the driver and the microchips when enough current went through the transistors before switching.

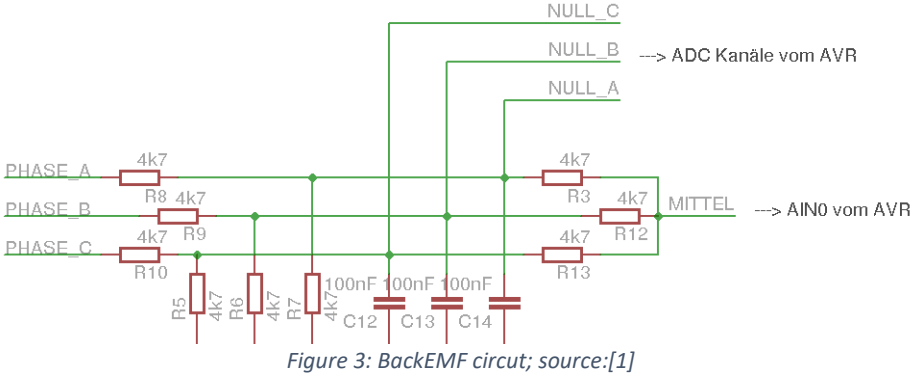

Figure 3: BackEMF circut; source:[1]

For measuring the BackEMF the circuit shown in figure 3 was adapted from [1]. Essentially the potential of every phase is divided to a level that can be feed to the microchip input ports. Capacitors are used to smoothen the voltage levels. A virtual neutral point was build using the resistors R3, R12 & R13. One of the three phases (depending on the current section) and the virtual neutral point must be compared, detecting zero crossings. In [1] this is done using the internal comparator of the microchip and an internal multiplexer to select between the phases. This lead to inaccurate measurement of the zero crossing, due to low cooperator accuracy of the internal comparator of the PIC chip in this project. An LM2902 op-amp for each phase is used to compare the two potentials reduce the comparator noise now, resulting in much better measurement of the zero-crossing. Also for each phase only the output of the op-amp must be feed to the microchip, resulting in one less pin used on the chip, comparing with the method of [1].

The values of the resistors of figure 3 were altered in comparison to [1], as a different dividing factor for the voltages is needed. For the resistors of the neutral point these smaller values were chosen, since the values in figure 1 had a too high input impedance for the internal comparator of the PIC.

A UA78M33C Voltage Regulator is used to provide a stable 3.3V level for the microchips. An electrolyte capacitor is used to smoothen VSS.

For a I2C master chip also the PIC16F1828 was used. Its I2C PIC's are connected to the ones of the motor driver. One of its analog inputs is used to read a potentiometer.

Figure 4 shows the whole circuit-board with the engine connected.

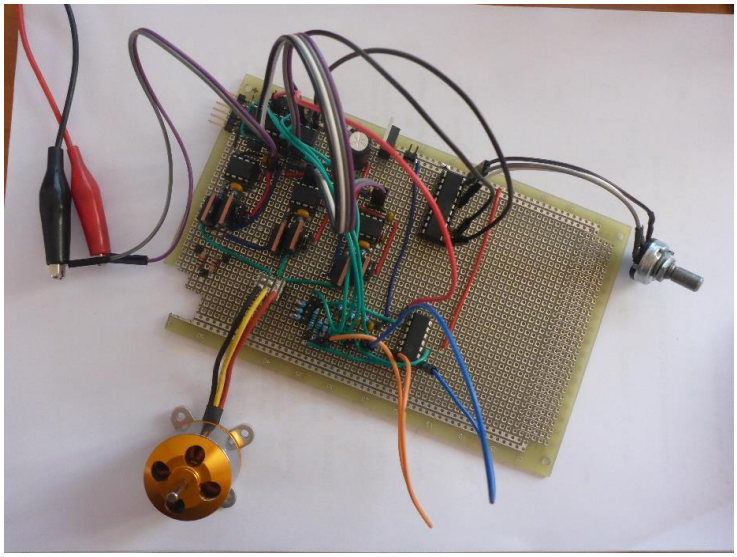

Figure 4: final circuit-board

Software:

The final software can be downloaded in [2] (commit 24.01.2018). The software in [2] could be subject to future change, since we will most likely continue working on the project.

The structure of the motor controller is quite easy. It contains the following functions:

void main(void):

After a device initialization a short period of forced commutation is used to set the motor into motion. Here commutation times are fixed intervals purely determined using the timer 0. Directly after that all phases are set to the floating state and the interrupt for the free commutation is turned on. This free commutation is handled purely in the interrupt routine. At first the free commutation is applied with

a fixed PWM duty cycle of 0.15% and in the field-weakening mode, where the commutation is directly triggered by the zero-crossing interrupt of the BackEMF. After a short time (determined by timer0) the mode is switched to the normal mode where the commutation in timed according to the speed of the motor using timers 4 and 6. After another while (determined by timer0) the I2C interface is switched on, allowing a master chip to command other values for the PWM duty circle. This procedure is implemented to ensure that the motor has reached a steady state before switching to the more fragile normal mode of free commutation and before accepting voltage commands. Finally, the I2C communication is handled in the main routine in an infinite while loop, since the commutation of the engine is handled by interrupts. Commutation is in this case more time-sensitive than the I2C, where the chip will simply stretch the clock while being in the interrupt.

void interrupt isr(void):

When an interrupt on change is detected indicating a zero crossing of the BackEMF in the right direction on the right phase timer 4 is used to measure and low-pass the current section length. After that if the field-weakening mode is turned on, the section is advanced by one. If the normal mode is in place, the timer 6 is set up to count town half the section length minus the approximated time needed to switch phases and turned on. If this time calculated for timer 6 is negative, the phase is advanced directly.

When a timer 6 interrupt is called, the phase is advanced by one.

In a previous iteration, the internal comparator was used to detect zero crossings of the BackEmf. An interrupt of the cooperator was used instead the interrupt on change and only the field-weakening mode was implemented, timer 6 was therefore not used.

void device_init(void):

This method is used for initialization of the oscillator, ports A, B & C, the PWM, the timers 0, 4, 6, the I2C interface for slave mode and interrupts on change of level of the outputs of the pins connected to the three phase's op-amps.

void setPWM(unsigned short pwm_cycle):

Here the PWM cycle length is set to pwm_cycle.

void setSection(unsigned int section):

Here the interrupt on change is switched off. The outputs to the half-bridge-drivers are set to drive the three phases according to the section. Afterwards the interrupt on change is set up to listen to the right edge of the right phase according to the section. Afterwards the interrupt is turned on again.

The software for the master chip is build the following:

void main(void):

After the device initialization in an infinite while loop it is waited for timer 0. Then the analog level of the potentiometer is read, a desired pwm value for the motor controller is calculated, and send to it via I2C. It is wait for the timer 0 for a second time. Finally, the current speed of the motor is requested via I2C. This speed value is currently not read, but requested to test the function and debug using a logic analyzer on the I2C lines.

void device_init(void):

Here the Initialization of the oscillator, all ports, the I2C module for master mode, the timer 0 and the analog digital converter (ADC) is implemented.

unsigned char getADCVal(void):

Here the voltage level of the potentiometer is read.

void i2cTransmit(unsigned char address, unsigned char data):

The method sends a write request to the slave with the address and writes the data byte to it if it receives an acknowledge from the address.

unsigned char i2cReceive(unsigned char address)

This method sends a read request to the slave with the address and requests one byte of data from the slave if an acknowledge was detected.

Results and Future Work

An effective motor controller for the motor was developed using a multitude of timers and interrupts working together. The controller can drive the BLDC engine in the field-weakening and the normal mode in a wide range of speeds while communicating over the I2C interface with a second chip. This master chip is commanding the motor controller effective voltage levels to put on the engine. The engine controller has a reliable way to determine the speed of the motor and can communicate this speed on the I2C interface when requested. With forced commutation a reliable way was found to start the motor.

The next step in the project will be to integrate a small feedback-controller for speed and a feedback controller for the current, which correlates with torque. For these steps it might be necessary to migrate the project to another microchip, since the PIC16f1828 has no hardware for floating point numbers and is currently at its computing limits when the engine is at high speeds. For the second controller a current-sensing unit will have to be implemented.

A better logic can be applied when calculating the motor speed eliminating outliers.

A circuit design and the use of SMD components will be essential to use the developed controller in an application.

References and Links:

[1] https://www.mikrocontroller.net/articles/Brushless-Controller_f%C3%BCr_Modellbaumotoren

[2] https://github.com/SRSonix/BLDC