# Microcontroller (MCU)

+5V
C1
0u1
GND

+5V
VCC 4

C2
4n7
GND

U2
ATtiny1617-M

RESET/PA0 — 23 RESET
PA1 — 24 SDA
PA2 — 1 SCL
PA3 — 2 PWM_IN
PA4 — 5 REVERSE
PA5 — 6 PWM5
PA6 — 7 A_BEMF
PA7 — 8 B_BEMF

PC0 — 17 ADD_1
PC1 — 18 ADD_2
PC2 — 19 ADD_3
PC3 — 20 PWM3
PC4 — 21 PWM4
PC5 — 22

PB0 — 16 PWM0
PB1 — 15 PWM1
PB2 — 14 TX
PB3 — 13 RX
PB4 — 12 C_BEMF
PB5 — 11 PWM2
PB6 — 10
PB7 — 9 ZERO

GND 3
GND

Short reverse pin to GND
reverse motor direction.
JP1
GND

Short these to ground to set
address of device on the I2C.
Addresses start at 0x20 and
go up to 0x27 (all shorted).

JP2
JP3
JP4
GND

PC5 is unavailable
on some variants

Responsible for controlling the entire system.
Recieves input and adjusts the motor speed accordingly.
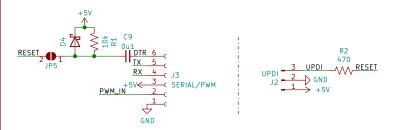Can also output current motor performance using digital communication.

Internal comparator is used for BEMF "zero"-cross detection.
A timer is used to create a time delayed event from the cross event
based on the time recorded between the cross and previous commutation.

# Control Connectors

Either I2C or PWM can be used to control this ESC.
I2C can act as both input and output for digital data.

PWM is standard for commercial ESCs. 1 to 2ms duty,
20ms period. PWM input is on the UART programming header.

+5V 4
SDA 3
SCL 2
1
J1
I2C
GND

# Programming Headers

+5V
D4
R1 10k
C9 0u1
RESET
JP5
DTR 6
TX 5
RX 4
+5V 3
PWM_IN 2
1
J3
SERIAL/PWM
GND

R2 470
UPDI 3
UPDI 2
1
J2
RESET
GND
+5V

The MCU is programmed by default using UPDI, but a bootloader can be
programmed so that it can be programmed like other Arduinos using UART.

There are connectors for both of these.

# Motor Control

+BATT
VDD 12

PWM2 1 HIN1
PWM1 2 LIN1
U1
FAN7888
PWM0 3 HIN2
PWM3 4 LIN2

PWM4 5 HIN3
PWM5 6 LIN3

GND 11
GND

VB1 20
HO1 19 AHO
VS1 18
LO1 17 ALO

VB2 16
HO2 15 BHO
VS2 14
LO2 13 BLO

VB3 10
HO3 9 CHO
VS3 8
LO3 7 CLO

D1 +BATT
C3 1u
A_OUT

D2 +BATT
C4 1u
B_OUT

D3 +BATT
C5 1u
C_OUT

RN1 20
AHO — 1    8 — AHO_F
ALO — 2    7 — ALO_F
BHO — 3    6 — BHO_F
BLO — 4    5 — BLO_F
CHO — 5    4 — CHO_F
CLO — 6    3 — CLO_F
7    2
8    1
20
RN2

+BATT
Q1
A_OUT
Q4
GND

+BATT
Q2
B_OUT
Q5
GND

+BATT
Q3
C_OUT
Q6
GND

MAX TOL:
20V
100A

FAN7888 MOSFET driver is used to drive the half-bridges for each phase of the motor based on control signals from the MCU.

N-channel MOSFETs are used on both sides of the bridge due to their lower "on" resistance. To drive the high-side FETs requires
"bootstrapping" to get a gate voltage higher than the drain (VBAT) for optimal performance.

Although the MOSFETs are rated for >100A, the traces are not! Add solder to increase current capacity

# BEMF Dividers

Back ElectroMotive Force (BEMF) is generated by a rapidly spinning motor. We monitor this on the floating phase
to determine when it is the right time to commute the motor to the next step.

The resistor network is used to divide the voltage down to a safe level (<5V) and derive a virtual zero by averaging
the voltage from each phase. These resistor networks need tight tolerances so the division ratios are as identical
as possible. Capacitors are there to help smooth out the signal, but are entirely optional.
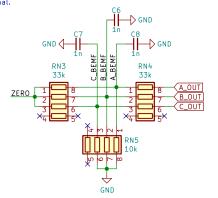
## Motor Control Therory

The basic theory for controlling a 3-phase BLDC motor is that when two
phases are connected to high and low, then the third floats [between them].

Let's say phase A is connected high, B is low, and C is left floating (not
connected to either power level) — but was previously high.

The floating voltage starts close the previous extreme it was connected to,
(high for C) then as the motor rotates through the step it will be pulled
toward the other extreme in a roughly linear fashion.

When this floating voltage crosses the average of all three phases ("zero"),
we are halfway through a commutation step, so we wait an equivalent
amount of time from the previous commutation to the zero crossing before
going to the next step. This is all handled by the MCU and its peripherals.

C6 1n GND
C7 1n GND
C8 1n GND
GND

RN3 33k
C_BEMF  B_BEMF  A_BEMF
ZERO
1    8
2    7
3    6
4    5
RN5 10k
RN4 33k
1    8 — A_OUT
2    7 — B_OUT
3    6 — C_OUT
4    5