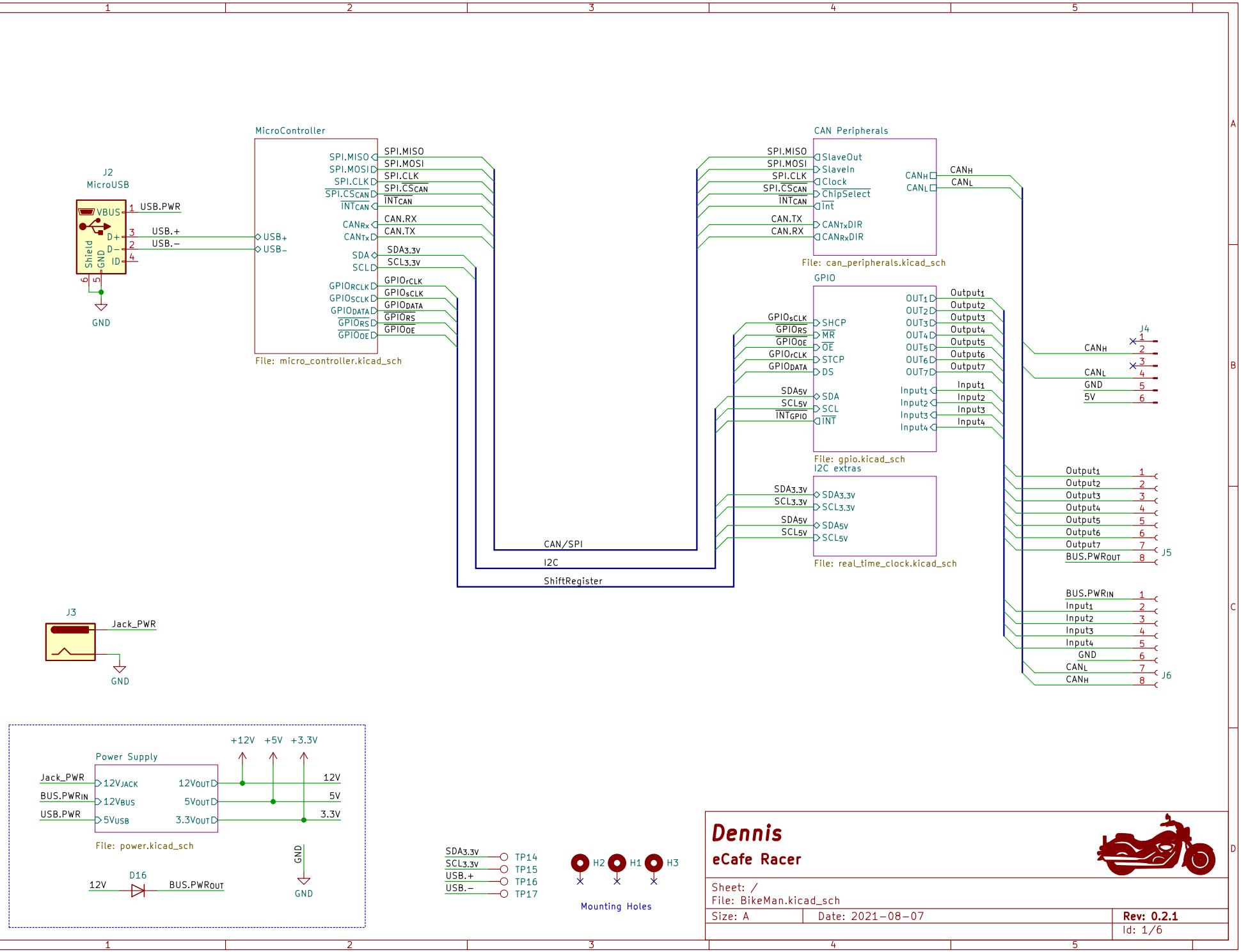
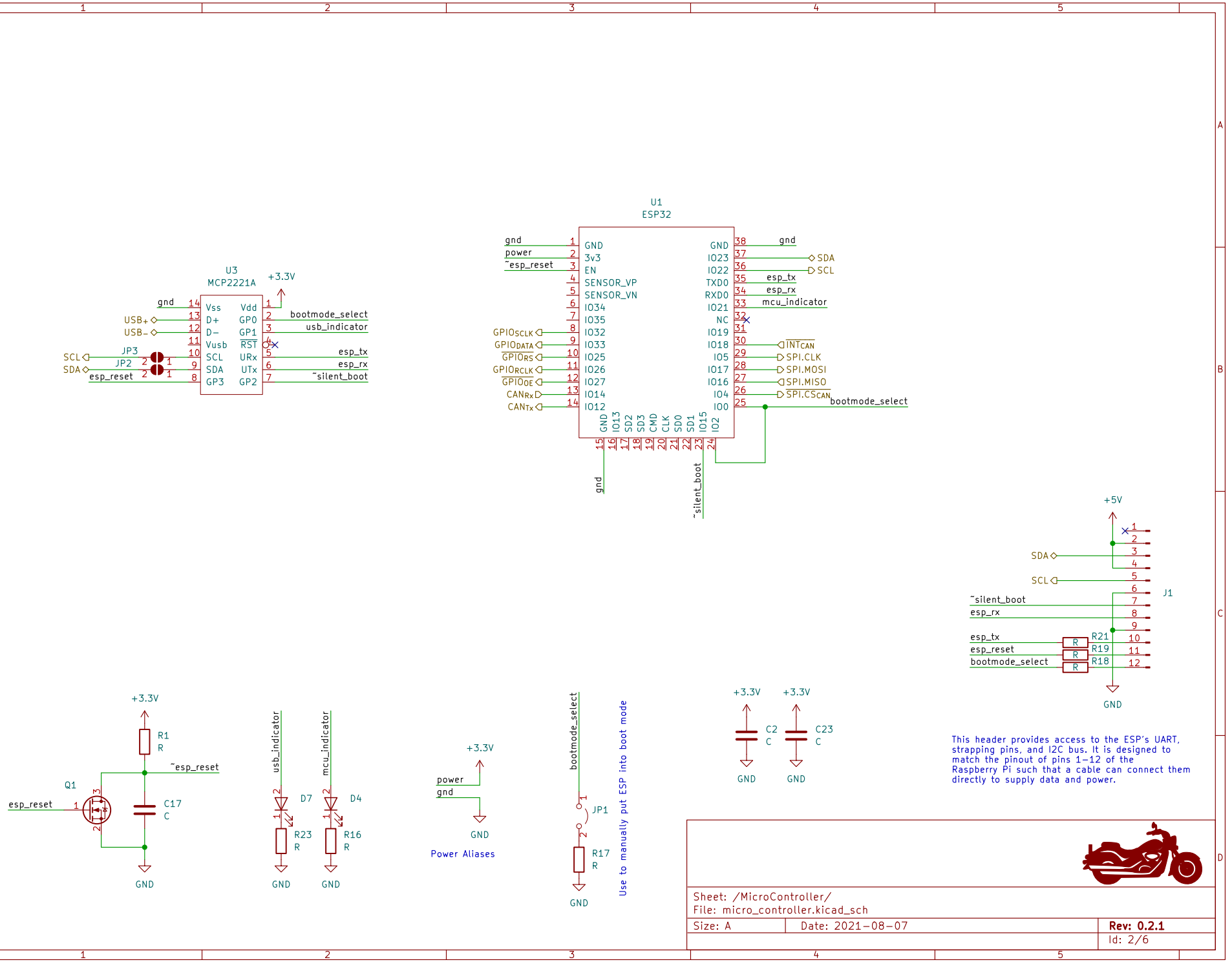
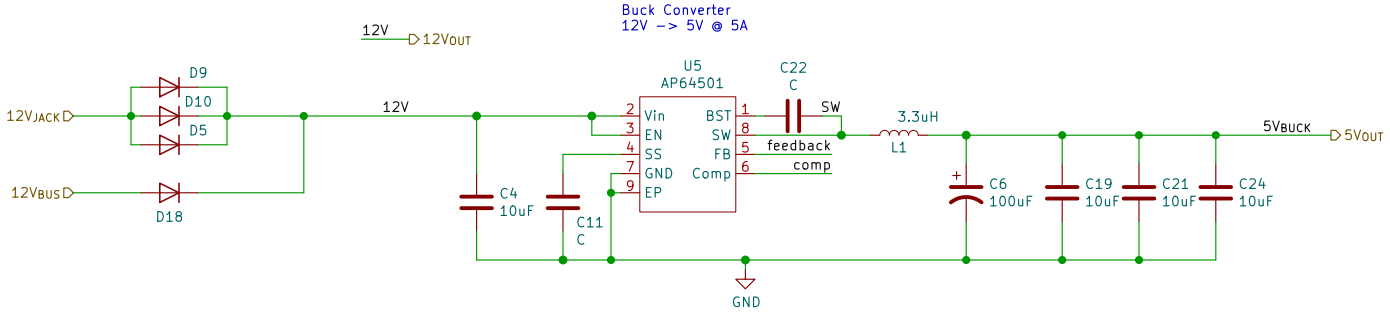
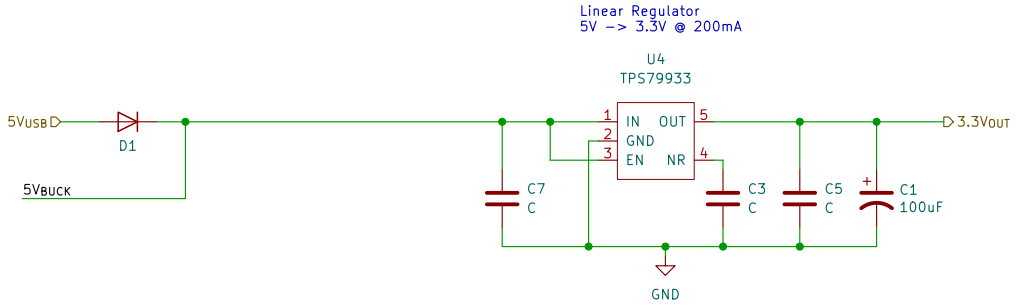
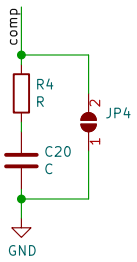

MicroController
J2
MicroUSB
VBUS
USB.PWR
USB.+
USB.−
Shield
GND
ID
USB+
USB−
SPI.MISO
SPI.MOSI
SPI.CLK
SPI.CSCAN
INTCAN
CANRx
CANTx
CAN.RX
CAN.TX
SDA
SCL
SDA3.3V
SCL3.3V
GPIORCLK
GPIOSCLK
GPIODATA
GPIORS
GPIOOE
File: micro_controller.kicad_sch
CAN Peripherals
SlaveOut
SlaveIn
Clock
ChipSelect
Int
CANTxDIR
CANRxDIR
CANH
CANL
File: can_peripherals.kicad_sch
GPIO
SHCP
MR
OE
STCP
DS
SDA
SCL
INT
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
Output1
Output2
Output3
Output4
Output5
Output6
Output7
Input1
Input2
Input3
Input4
SDA5V
SCL5V
INTGPIO
File: gpio.kicad_sch
I2C extras
SDA3.3V
SCL3.3V
SDA5V
SCL5V
File: real_time_clock.kicad_sch
CAN/SPI
I2C
ShiftRegister
J4
J5
J6
BUS.PWROUT
BUS.PWRIN
5V
J3
Jack_PWR
Power Supply
+12V
+5V
+3.3V
12VJACK
12VBUS
5VUSB
12VOUT
5VOUT
3.3VOUT
12V
5V
3.3V
File: power.kicad_sch
D16
BUS.PWROUT
TP14
TP15
TP16
TP17
H1
H2
H3
Mounting Holes
Dennis
eCafe Racer
Sheet: /
File: BikeMan.kicad_sch
Size: A
Date: 2021-08-07
Rev: 0.2.1
Id: 1/6


U3
MCP2221A
+3.3V
gnd
USB+
USB−
JP3
JP2
SCL
SDA
esp_reset
Vss
D+
D−
Vusb
SCL
SDA
GP3
Vdd
GP0
GP1
RST
URx
UTx
GP2
bootmode_select
usb_indicator
esp_tx
esp_rx
~silent_boot
U1
ESP32
gnd
power
~esp_reset
GPIOSCLK
GPIODATA
GPIORS
GPIORCLK
GPIOOE
CANRx
CANTx
GND
3v3
EN
SENSOR_VP
SENSOR_VN
IO34
IO35
IO32
IO33
IO25
IO26
IO27
IO14
IO12
IO13
SD2
SD3
CMD
CLK
SD0
SD1
IO15
IO2
IO23
IO22
TXD0
RXD0
IO21
NC
IO19
IO18
IO5
IO17
IO16
IO4
IO0
SDA
SCL
esp_tx
esp_rx
mcu_indicator
INTCAN
SPI.CLK
SPI.MOSI
SPI.MISO
SPI.CSCAN
bootmode_select
+5V
J1
~silent_boot
esp_rx
esp_tx
esp_reset
bootmode_select
R21
R19
R18
R
GND
+3.3V
R1
R
~esp_reset
Q1
esp_reset
C17
C
GND
usb_indicator
mcu_indicator
D7
D4
R23
R16
R
GND
GND
+3.3V
power
gnd
GND
Power Aliases
bootmode_select
JP1
R17
R
GND
Use to manually put ESP into boot mode
+3.3V
+3.3V
C2
C23
C
GND
GND
This header provides access to the ESP's UART, strapping pins, and I2C bus. It is designed to match the pinout of pins 1–12 of the Raspberry Pi such that a cable can connect them directly to supply data and power.
Sheet: /MicroController/
File: micro_controller.kicad_sch
Size: A
Date: 2021-08-07
Rev: 0.2.1
Id: 2/6

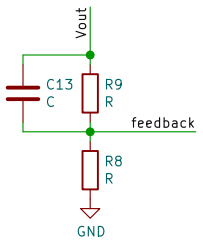






Compensation Circuit

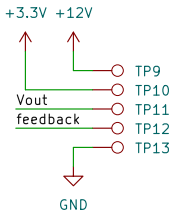
*See Datasheet

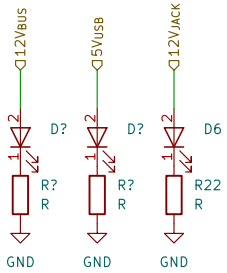


Feedback Circuit

Vout is determined by this feedback divider circuit with the equation:
$R_{high} = R_{low} * (V_{out} / 0.8V - 1)$

*C4 is optional for improving transient response
*Use at least 1% precision resistors





Power Indicators

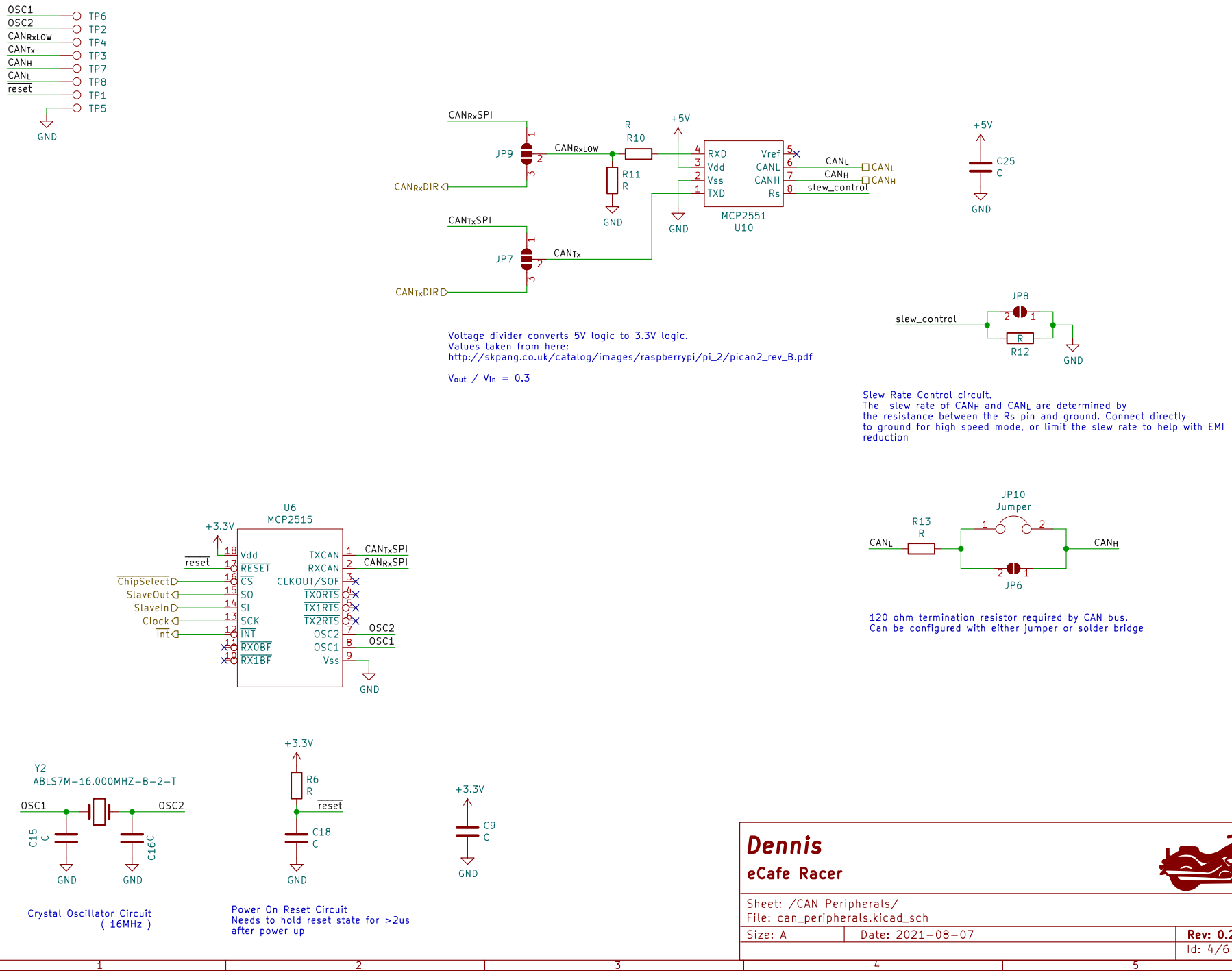
Dennis uses 3 supply voltages:
12V, 5V, and 3.3V

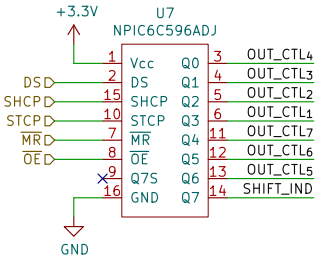
Dennis

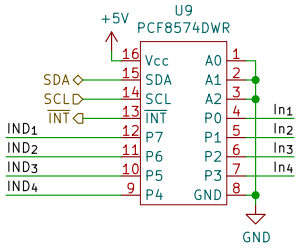
eCafe Racer

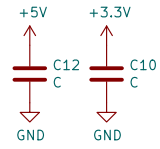
Sheet: /Power Supply/
File: power.kicad_sch

Size: A	Date: 2021-08-07	**Rev: 0.2.1**
		Id: 3/6

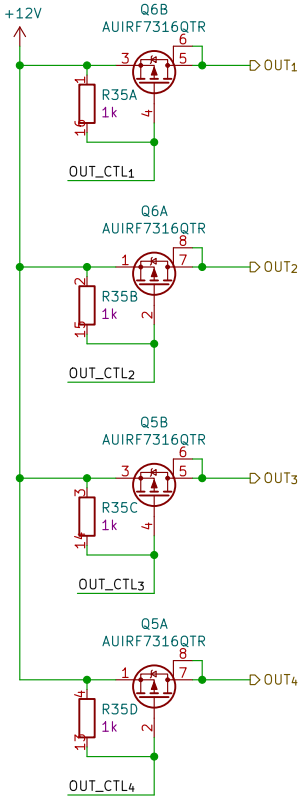

OSC1
OSC2
CANRxLOW
CANTx
CANH
CANL
reset
TP6
TP2
TP4
TP3
TP7
TP8
TP1
TP5
GND
CANRxSPI
JP9
CANRxLOW
R
R10
+5V
R11
R
GND
CANRxDIR
RXD
Vdd
Vss
TXD
Vref
CANL
CANH
Rs
MCP2551
U10
CANL
CANH
slew_control
CANTxSPI
JP7
CANTx
CANTxDIR
+5V
C25
C
GND
JP8
slew_control
R
R12
GND
Voltage divider converts 5V logic to 3.3V logic.
Values taken from here:
http://skpang.co.uk/catalog/images/raspberrypi/pi_2/pican2_rev_B.pdf
Vout / Vin = 0.3
Slew Rate Control circuit.
The slew rate of CANH and CANL are determined by
the resistance between the Rs pin and ground. Connect directly
to ground for high speed mode, or limit the slew rate to help with EMI
reduction
U6
MCP2515
+3.3V
reset
ChipSelect
SlaveOut
SlaveIn
Clock
Int
Vdd
RESET
CS
SO
SI
SCK
INT
RX0BF
RX1BF
TXCAN
RXCAN
CLKOUT/SOF
TX0RTS
TX1RTS
TX2RTS
OSC2
OSC1
Vss
CANTxSPI
CANRxSPI
OSC2
OSC1
GND
JP10
Jumper
R13
R
CANL
CANH
JP6
120 ohm termination resistor required by CAN bus.
Can be configured with either jumper or solder bridge
Y2
ABLS7M-16.000MHZ-B-2-T
OSC1
OSC2
C15
C
C16C
GND
GND
Crystal Oscillator Circuit
(16MHz)
+3.3V
R6
R
reset
C18
C
GND
Power On Reset Circuit
Needs to hold reset state for >2us
after power up
+3.3V
C9
C
GND
Dennis
eCafe Racer
Sheet: /CAN Peripherals/
File: can_peripherals.kicad_sch
Size: A
Date: 2021-08-07
Rev: 0.2.1
Id: 4/6

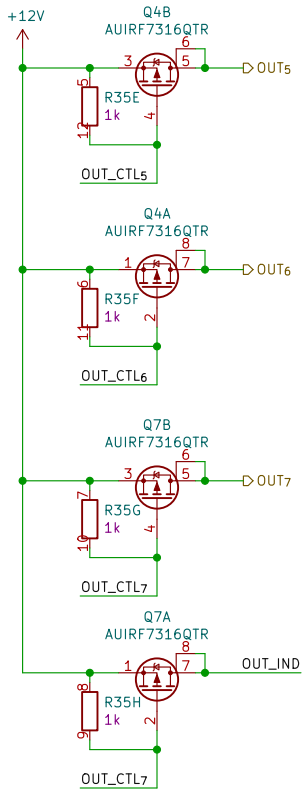


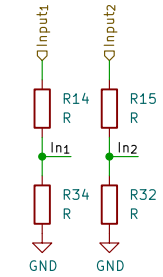




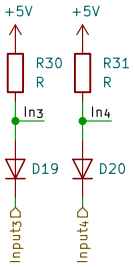
An I2C IO expander is used to read the inputs instead of connecting to the pi's gpio pins directly. This is mainly done for insurance so that the pi's pins cannot be accidentally exposed to the 12V signal.





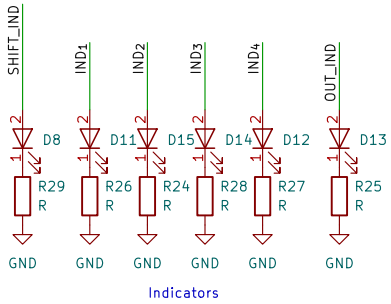


Voltage Divider Network
(1 / 3 : 12V -> 4V)
Used to read 12V digital signals
from other parts of the bike



Switch To Ground Detector
Used to read state of simple
mechanical switches on the bike.

Diodes prevent failure from
miswiring of the two input types



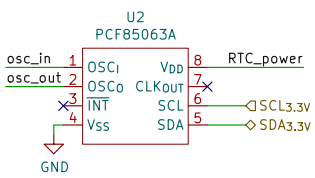
Indicators

By far the most common use for these inputs will be to read the value of switches. Need to think more about the best way to achieve this, it is most likely not this.

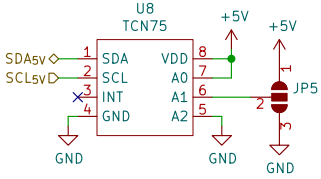
It might be nice to combine Input and Output pins into a single circuit which can be configured in software to act like either, like the gpio modules on microprocessors. Perhaps a "switch detection" mode could be added as well, which measures the resistance to ground instead of voltage

Dennis
eCafe Racer

Sheet: /GPIO/		
File: gpio.kicad_sch		
Size: A	Date: 2021-08-07	**Rev: 0.2.1**
		Id: 5/6

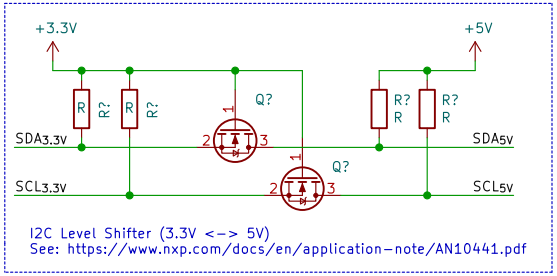


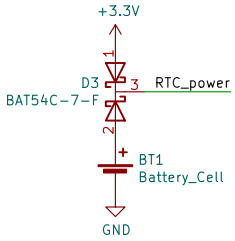
Real Time Clock

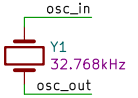


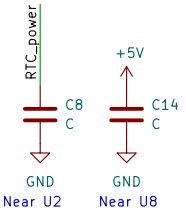
Address Select Jumper

Ambient Temp Sensor









Dennis

eCafe Racer

Sheet: /I2C extras/
File: real_time_clock.kicad_sch

Size: A	Date: 2021-08-07	**Rev: 0.2.1**
		Id: 6/6