circulator

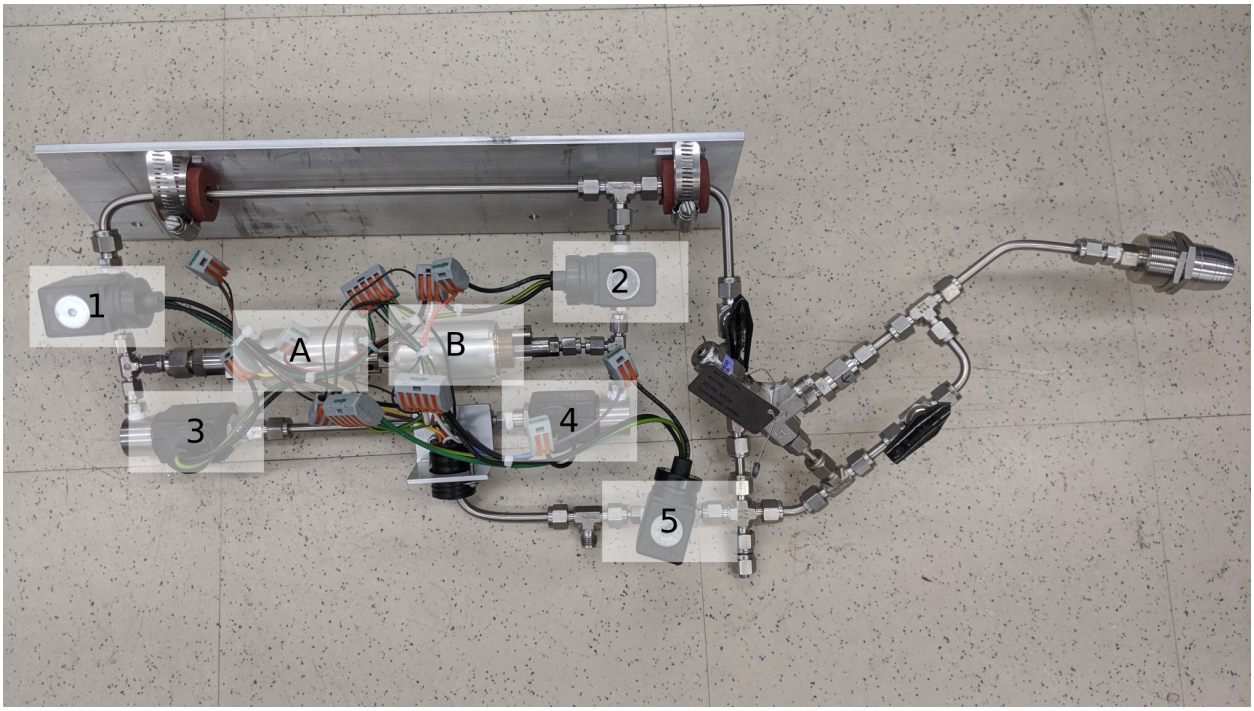
November 11, 2019

1 description

The circulator is responsible for mass transport within the WiHP-NMRR. It consists of five solenoid valves, and two core solenoids that drive a central piston. These fire in a certain pattern to drive gas through the WiHP-NMRR.

The circulator driver is an box of electronics that drive the seven solenoids.

2 solenoids, cable



The image above shows the circulator itself, with the five solenoid valves and two piston solenoids labeled. The valves are labeled with numbers, one through five. The piston drivers are labeled with letters, A and B. This convention is maintained throughout this document.

Valves 1, 2 and 5 are normally closed. Valves 3 and 4 are normally open. The following pattern must be followed to circulate:

	"tick"	"tock"
A	energized	denergized
2	energized (open)	denergized (closed)
4	energized (closed)	denergized (open)
B	denergized	energized
1	denergized (closed)	energized (open)
3	denergized (open)	energized (closed)

On the "tick" step, the piston moves from B to A, pulling gas through valve 2 and pushing gas out through valve 3. On the "tock" step, the piston moves from A to B, pulling gas through valve 1 and pushing gas out through valve 4.

Valve five bypasses the circulator, and is operated separately from the rest of the solenoids. The circulator is bypassed when valve five is energized.

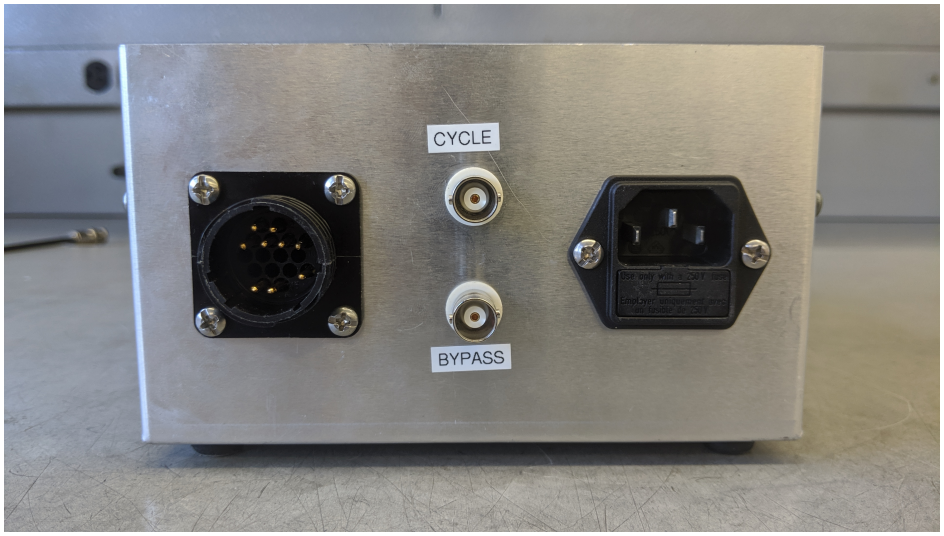
Electrically, the five valves all operate on 120 V AC. The two piston solenoids operate at 12 V DC. Each of the components has a separate wire controlling line (or positive) voltage within the cable. The components share ground and neutral. The piston solenoids are ground referenced.

The circulator is connected by a circular plastic connector, TE/AMP part numbers 206037-1 and 206036-1. The pin designations are as follows.

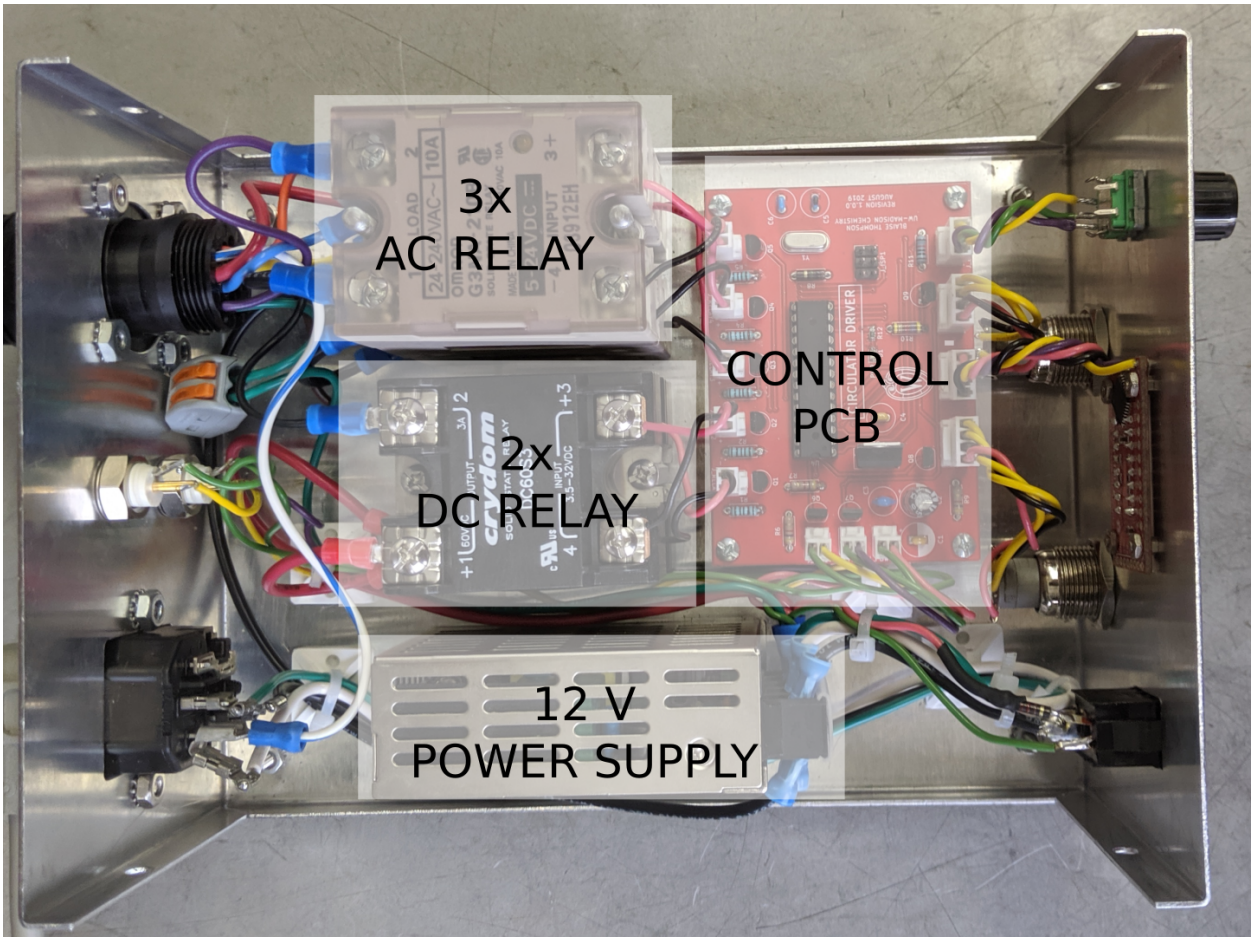
pin	color	assignment
1	white	neutral
2	red	valve 1
3	orange	valve 2
4	yellow	valve 3
5	blue	valve 4
6	purple	valve 5
7		
8		
9		
10		
11		
12		
13		
14	green	ground
15	brown	solenoid A
16	black	solenoid B

3 electrical box

The circulator driver controls the firing of the valves and solenoids necessary for mass transport in the WiHP-NMRR. The image above is the front of the circulator driver box. There is a power switch, far left. The seven segment display in the middle shows the current circulator frequency, in Hz. This frequency can be adjusted using the encoder knob, right. Two buttons, one for bypass and the other for cycle, control the current status of the circulator. Mechanically these are toggle buttons, and each toggles their corresponding state. Each button contains an illuminated ring which will always display the current state. These states can also be updated via TTL, as will be discussed later.



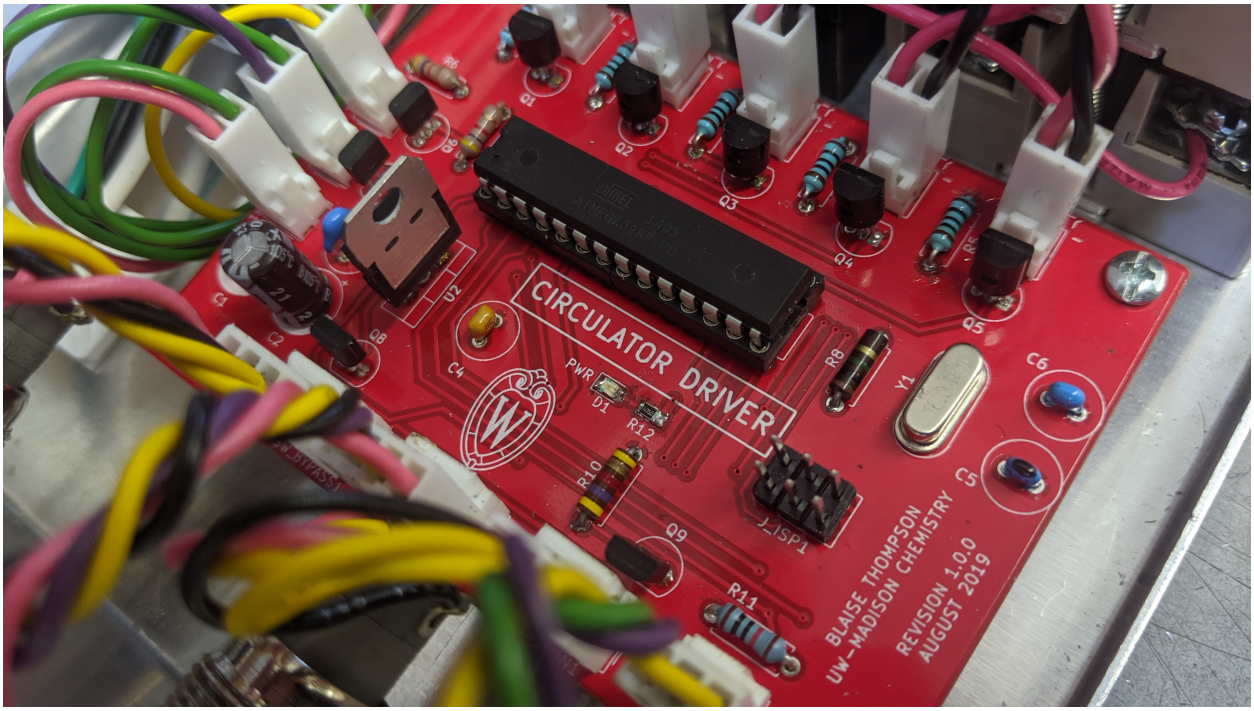
The image above shows the rear of the circulator driver box. There is a sixteen-pin CPC connector, left, which goes to the circulator itself. In the middle there are two BNC inputs, cycle and bypass, which accept TTL control from the NMR. On the right there is power input, which accepts a standard C13 connector. The circulator operates at 120 V, and may draw up to 5 amps. There is a fuse within the drawer immediately under the AC entry port.



The image above shows the internals of the circulator driver box. The enclosure used is 8" x 6" by 3.5", Bud Industries part number CU-3009-A. Besides the front and back panel elements, the box contains a 12 V power supply (TRACO part number TXL 025-12S), 2 DC relays (Crydom part number DC60S3), and 3 AC relays (Omron part number G3NA-205B). The relays and power supply make up the majority of the cost of the drive electronics.

Also contained within the enclosure is the custom control printed circuit board (PCB). This central control board accepts power from the 12 V supply, input from the two rear-panel BNC connectors, and connects to the front panel elements and five relays. At the core of the control board is a microcontroller with custom firmware. These will all be discussed in the next few sections of this manual.

4 circuit board



The following pages diagram the printed circuit board (PCB) at the heart of the circulator driver electronics. This PCB is centered around a single microcontroller, the ATmega328. A 16 MHz crystal supplies the clock for the microcontroller. A standard six pin ISP header can be used to program the microcontroller in-circuit.

The microcontroller has outputs to drive five relays, each controlling the action of a small NPN transistor (2N7000) with an appropriate pull-up resistor. These five relays drive the circulator solenoids, as described in the previous section. The microcontroller has two TTL inputs, each buffered by a small PNP transistor (BS250) and associated pull-down resistor.

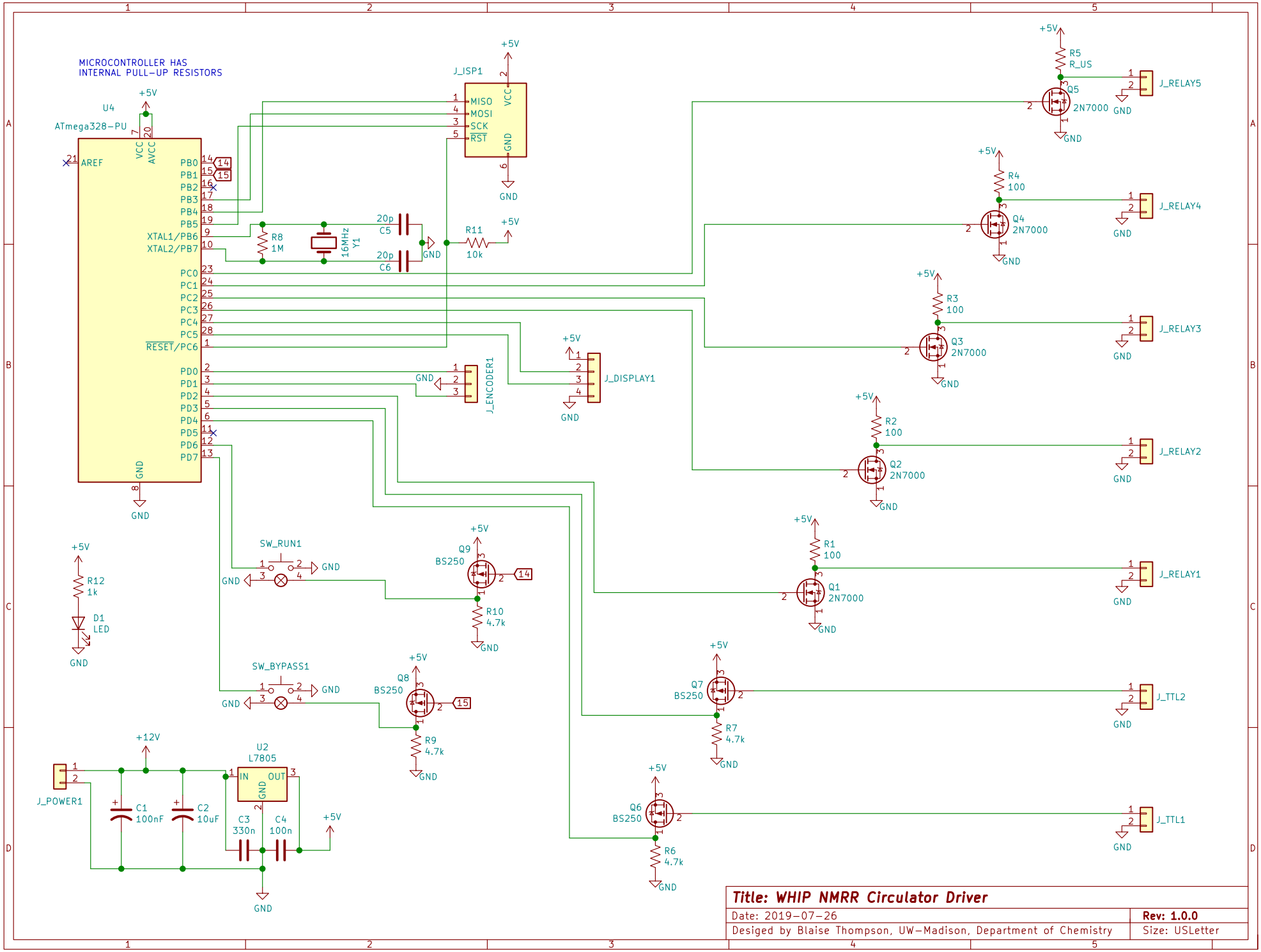
Besides these, the microcontroller has inputs for the two front-panel buttons and the front-panel encoder. There are also outputs for the LEDs embedded within each front-panel button, and for the seven-segment display (SparkFun part number COM-11441) (driven via I2C).

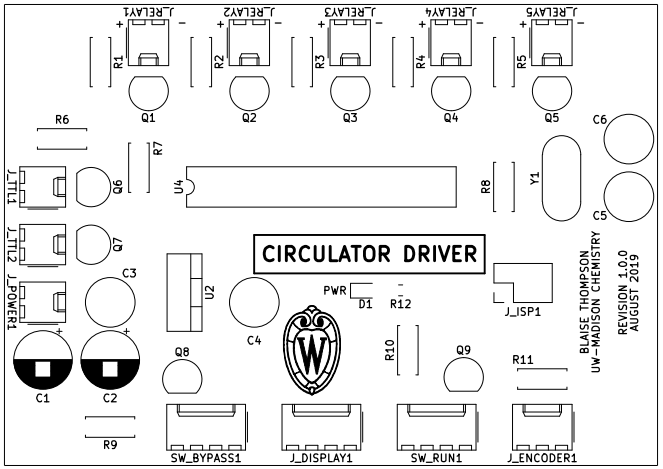
The following pages contain:

1. PCB schematic
2. PCB silk screen
3. PCB top copper layer
4. PCB bottom copper layer

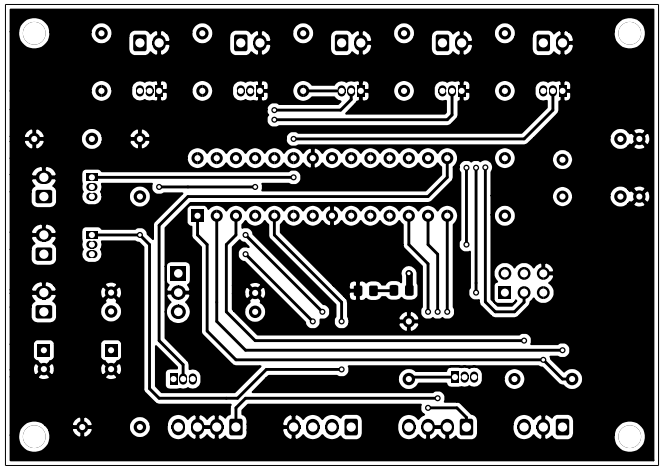
Please note that PCB silk screen and copper representations are to-scale.

The next sections describe the microcontroller firmware and PCB components.

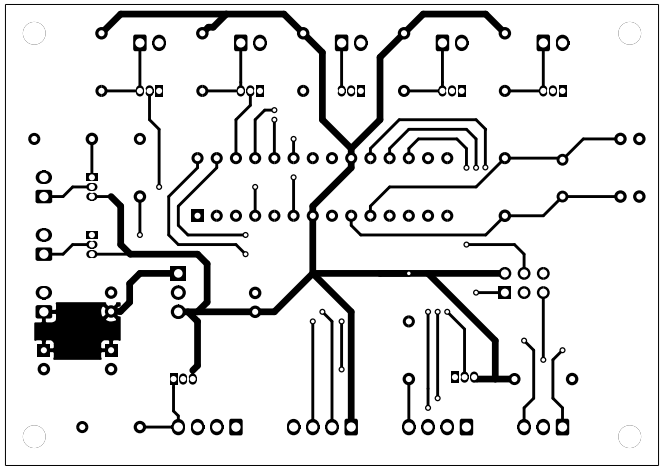

MICROCONTROLLER HAS
INTERNAL PULL-UP RESISTORS
U4
ATmega328-PU
AREF
VCC
AVCC
PB0
PB1
PB2
PB3
PB4
PB5
XTAL1/PB6
XTAL2/PB7
PC0
PC1
PC2
PC3
PC4
PC5
RESET/PC6
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
GND
+5V
J_ISP1
MISO
MOSI
SCK
RST
R8
1M
16MHz
Y1
20p
C5
20p
C6
R11
10k
J_ENCODER1
J_DISPLAY1
R5
R_US
Q5
2N7000
J_RELAY5
R4
100
Q4
2N7000
J_RELAY4
R3
100
Q3
2N7000
J_RELAY3
R2
100
Q2
2N7000
J_RELAY2
R1
100
Q1
2N7000
J_RELAY1
SW_RUN1
Q9
BS250
R10
4.7k
SW_BYPASS1
Q8
BS250
R9
4.7k
Q7
BS250
R7
4.7k
J_TTL2
Q6
BS250
R6
4.7k
J_TTL1
R12
1k
D1
LED
J_POWER1
+12V
U2
L7805
IN
OUT
C1
100nF
C2
10uF
C3
330n
C4
100n
Title: WHIP NMRR Circulator Driver
Date: 2019-07-26
Rev: 1.0.0
Desiged by Blaise Thompson, UW-Madison, Department of Chemistry
Size: USLetter



Title: WHIP-NMRR Circulator Driver	
Date: 2019-07-26	Rev: 1.0.0
Desiged by Blaise Thompson, UW-Madison, Department of Chemistry	Size: USLetter



Title: WHIP-NMRR Circulator Driver	
Date: 2019-07-26	Rev: 1.0.0
Desiged by Blaise Thompson, UW-Madison, Department of Chemistry	Size: USLetter



Title: WHIP-NMRR Circulator Driver	
Date: 2019-07-26	Rev: 1.0.0
Desiged by Blaise Thompson, UW-Madison, Department of Chemistry	Size: USLetter

5 firmware

The following pages contain the raw source-code running on the embedded Atmega328 microcontroller.

```
// firmware: circulator-driver

// 7SEG display (I2C communication)
#include <Wire.h> // Include the Arduino SPI library
const byte s7sAddress = 0x71;
char tempString[10];  // used with sprintf to create strings

// rotary encoder
#include <Encoder.h>
Encoder knob(1, 0);  // interrupt, registered in library
long frequency  = 2000;  // mHz

// control pins
int relay1 = 2;  // physical pin 4
int relay2 = A3;  // physical pin 26
int relay3 = A2;  // physical pin 25
int relay4 = A1;  // physical pin 24
int relay5 = A0;  // physical pin 23

// buttons
int bypass_button = 7;  // physical pin 13
int bypass_LED = 9;  // physical pin 15
int run_button = 6;  // physical pin 12
int run_LED = 8;  // physical pin 14

// TTL
int TTL1 = 4;  // physical pin 6
int TTL1_state = HIGH;
int TTL2 = 3;  // physical pin 5
int TTL2_state = HIGH;

// run control
bool bypass_state = LOW;
bool run_state = LOW;

// timing control
bool state = LOW;
float period = 1000;           // interval at which to blink (milliseconds)
unsigned long previousMillis = 0;        // will store last time LED was updated

void setup()
{
  // 7SEG display (I2C communication)
  Wire.begin();  // Initialize hardware I2C pins
  clearDisplayI2C();  // Clears display, resets cursor
  s7sSendStringI2C("-HI-");
  setDecimalsI2C(0b111111);  // Turn on all decimals, colon, apos
  setBrightnessI2C(255);  // 0 to 255
  clearDisplayI2C();
```

```
  setDecimalsI2C(0b00000010);
  sprintf(tempString, "%04d", frequency / 10);
  s7sSendStringI2C(tempString);
  // rotary encoder
  knob.write(0);
  // serial
  Serial.begin(9600);
  // output pins
  pinMode(relay1, OUTPUT);
  pinMode(relay2, OUTPUT);
  pinMode(relay3, OUTPUT);
  pinMode(relay4, OUTPUT);
  pinMode(relay5, OUTPUT);
  pinMode(bypass_LED, OUTPUT);
  pinMode(run_LED, OUTPUT);
  // input pins
  pinMode(bypass_button, INPUT_PULLUP);
  pinMode(run_button, INPUT_PULLUP);
  pinMode(TTL1, INPUT_PULLUP);
  pinMode(TTL2, INPUT_PULLUP);
  //
  delay(1000);
}

void loop() {
  // buttons (active low)
  if (digitalRead(bypass_button) == LOW) {
    while(digitalRead(bypass_button) == LOW) {
      delay(1);  // ms
    }
    bypass_state = !bypass_state;
  }
  digitalWrite(bypass_LED, !bypass_state);  // active low
  if (digitalRead(run_button) == LOW) {
    while(digitalRead(run_button) == LOW) {
      delay(1);  // ms
    }
    run_state = !run_state;
  }
  // TTL
  int ttl1 = digitalRead(TTL1);
  if (ttl1 != TTL1_state) {
    TTL1_state = ttl1;
    run_state = !ttl1;
  }
  int ttl2 = digitalRead(TTL2);
  if (ttl2 != TTL2_state) {
    TTL2_state = ttl2;
    bypass_state = !ttl2;
    run_state = ttl2;
  }
  digitalWrite(run_LED, !run_state);  // active low
  // input from rotary encoder
  int x = knob.read();
```

```
  int sign = (x > 0) - (x < 0);
  if (abs(x) >= 4) {
    if (frequency == 25000) {
      if (sign == -1) {
        frequency -= 1000;
      }
    }
    else if (10000 < frequency) {
      frequency += 1000 * sign;
    }
    else if (frequency == 10000) {
      if (sign == 1) frequency += 1000;
      if (sign == -1) frequency -= 100;
    }
    else if ((1000 < frequency) && (frequency < 10000)) {
      frequency += 100 * sign;
    }
    else if (frequency == 1000) {
      if (sign == 1) frequency += 100;
      if (sign == -1) frequency -= 10;
    }
    else if ((20 <= frequency) && (frequency < 1000)) {
      frequency += 10 * sign;
    }
    else{
      if (sign == 1) {
        frequency += 10;
      }
    }
    sprintf(tempString, "%04d", frequency / 10);
    s7sSendStringI2C(tempString);
    knob.write(0);
    period = 1e6 / (float)frequency;
  }
  // control cycle relays
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= (period / 2)) {
    previousMillis = currentMillis;
    if (run_state == HIGH) {
      state = !state;
    }
    digitalWrite(relay1, state);  // A
    digitalWrite(relay4, state);  // 2, 4
    digitalWrite(relay2, !state);  // B
    digitalWrite(relay3, !state);  // 1, 3
  }
  // control bypass relay
  digitalWrite(relay5, !bypass_state);  // low active
}

void s7sSendStringI2C(String toSend) {
  // This custom function works somewhat like a serial.print.
  //  You can send it an array of chars (string) and it'll print
  //  the first 4 characters in the array.
```

```
  Wire.beginTransmission(s7sAddress);
  for (int i=0; i<4; i++) {
    Wire.write(toSend[i]);
  }
  Wire.endTransmission();
}

void clearDisplayI2C() {
  // Send the clear display command (0x76)
  //  This will clear the display and reset the cursor
  Wire.beginTransmission(s7sAddress);
  Wire.write(0x76);  // Clear display command
  Wire.endTransmission();
}

void setBrightnessI2C(byte value) {
  // Set the displays brightness. Should receive byte with the value
  //  to set the brightness to
  //  dimmest------------->brightest
  //     0--------127--------255
  Wire.beginTransmission(s7sAddress);
  Wire.write(0x7A);  // Set brightness command byte
  Wire.write(value);  // brightness data byte
  Wire.endTransmission();
}

void setDecimalsI2C(byte decimals) {
  // Turn on any, none, or all of the decimals.
  //  The six lowest bits in the decimals parameter sets a decimal
  //  (or colon, or apostrophe) on or off. A 1 indicates on, 0 off.
  //  [MSB] (X)(X)(Apos)(Colon)(Digit 4)(Digit 3)(Digit2)(Digit1)
  Wire.beginTransmission(s7sAddress);
  Wire.write(0x77);
  Wire.write(decimals);
  Wire.endTransmission();
}
```

6 BOM

The following is an INCOMPLETE bill of materials for the circulator. Trivial components like screws and resistors have been excluded.

item	part	rate	count	total	comment
AC relay	G3NA-205B	27.00	3	81.00	
DC relay	DC60S3	21.00	2	42.00	
12 V power supply	TXL 025-12S	37.00	1	37.00	
enclosure	CU-3009-A	16.00	1	16.00	
CPC pin	66103-4	0.75	20	15.00	
CPC socket	1-66105-9	0.75	20	15.00	
CPC plug	206037-1	6.00	2	12.00	
CPC socket	206036-1	5.00	2	10.00	
BNC jack	031-10-RFXG1	3.00	2	6.00	
AC entry	6200-2300	6.00	1	5.00	
16 MHz xtal	HC-49/U-S16000000ABJB	1.00	1	1.00	
header, 2 pin	22-23-2021	0.25	8	2.00	
header, 4 pin	22-23-2041	0.50	3	1.50	
header, 3 pin	22-23-2031	0.25	1	0.25	
DIP socket, 14 pin	110-93-314-41-0010000	1.50	2	3.00	
housing, 2 pin	22-01-3027	0.25	8	2.00	
housing, 3 pin	22-01-3037	0.50	1	0.50	
mosfet, N-channel	2N7000	0.25	5	1.25	
fuse, 1 A fast		1.00	2	2.00	
ring terminal, #6, 14-16 AWG	34158	0.50	10	5.00	
capacitor, 33 nF	FG14X7R1H334KNT06	0.25	1	0.25	
standoff, 0.5" #4-40		0.50	4	2.00	
housing, 4 pin	22-01-3047	0.50	3	1.50	
capacitor, 100 uF		0.25	1	0.25	
regulator, 5 V	7805	1.00	1	1.00	
switch	R1966ABLKBLKEFGRN	2.00	1	2.00	
7 segment display	COM-11441	15.00	1	15.00	
rotary encoder	EN11	1.50	1	1.50	
button, illuminated		2.00	2	4.00	
terminal block, 2 contact	222-412	0.50	8	4.00	
terminal block, 3 contact	222-413	0.50	2	1.00	
terminal block, 5 contact	222-415	0.75	3	2.25	
rubber bumper	720	0.25	4	1.00	