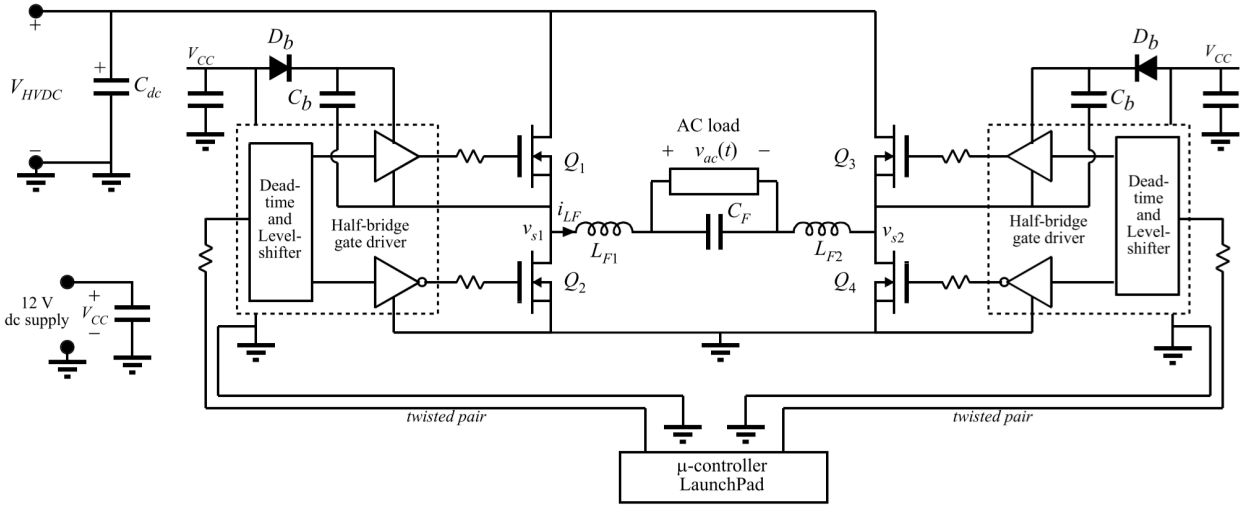
Experiment #5
DC-AC Inverter

Dane Thorn and Andrew Thibeault
ECEN 5517
2023-04-24

Introduction:

The objective of this experiment is to design, construct, test, and demonstrate a single-phase dc-ac inverter powered from the closed-loop regulated step-up dc-dc converter designed in experiment 4. We developed an H-bridge inverter, and controlled it to produce a modified sine-wave, 120VRMS, 60Hz signal. This wave-form is rectangular, and can be seen in Figure 5. A true sine-wave output was designed subsequently by varying the duty cycle sinusoidally.



Step 1: Modified Sine-Wave Inverter Circuit Diagram

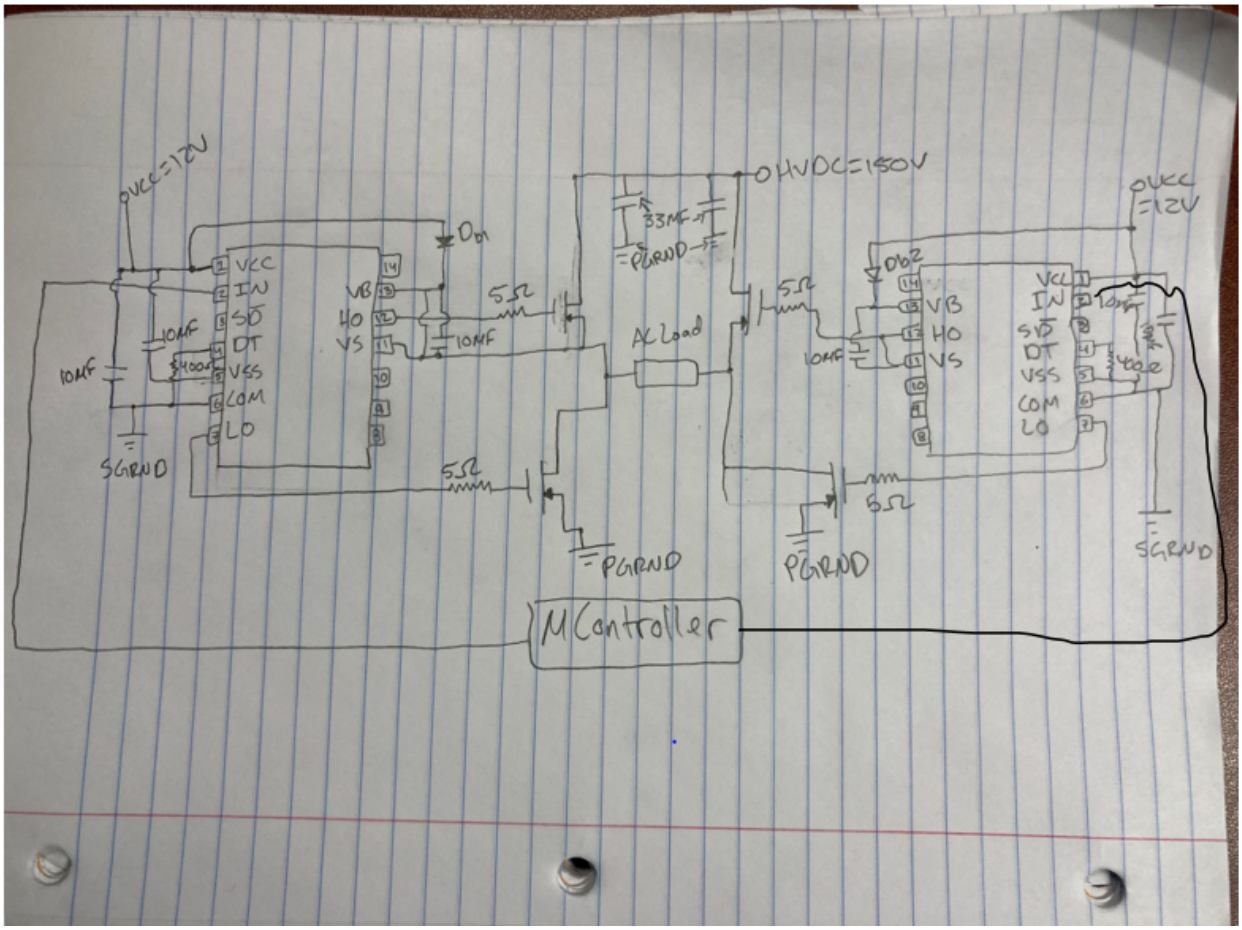


Figure 1: Circuit Diagram of Modified Sine-Wave Inverter

Step 2: Modified Sine-Wave Inverter Circuitry

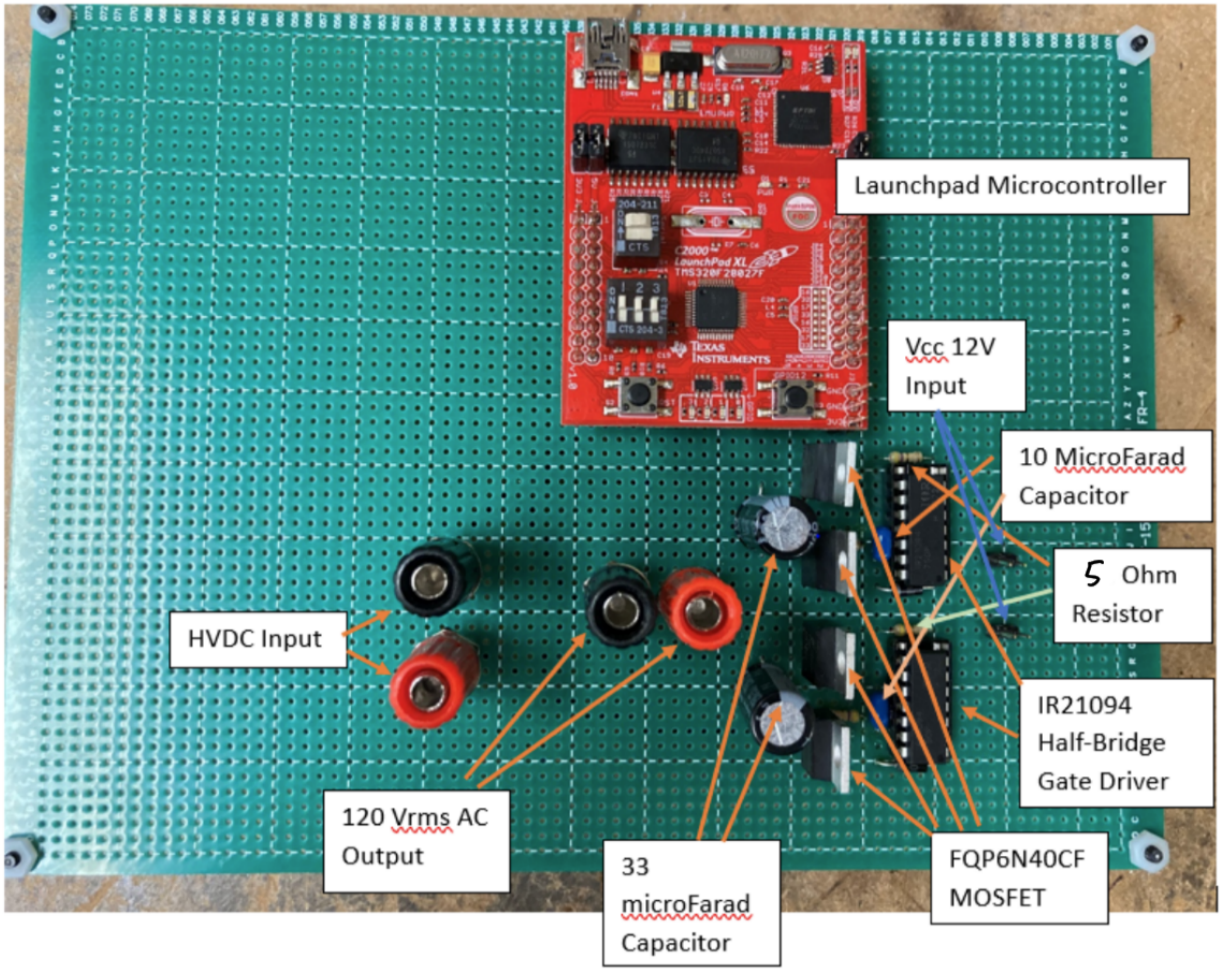


Figure 2: Implemented Construction of the Modified Sine-Wave Inverter

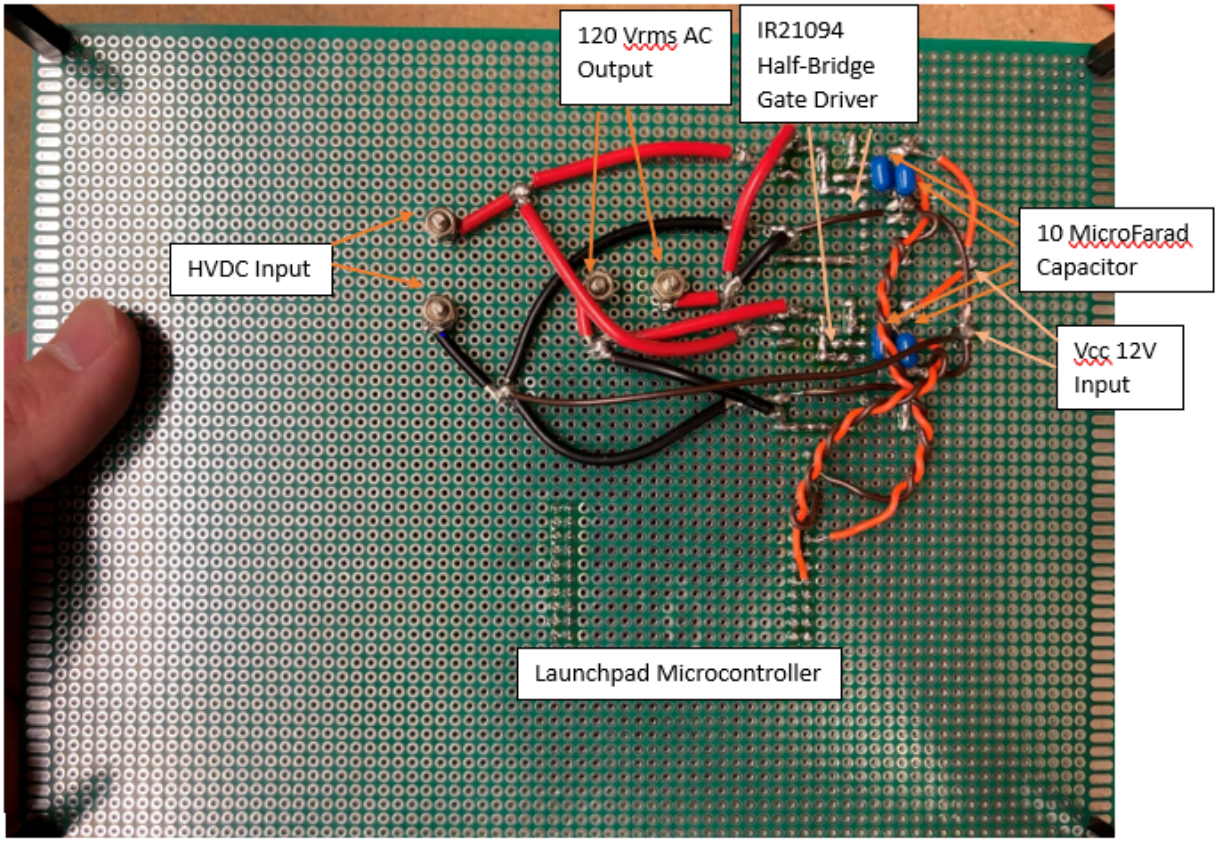


Figure 3: Bottom side of the Implemented Construction of the Modified Sine-Wave Inverter

Step 3: Modified Sine-Wave Gate Control Signals

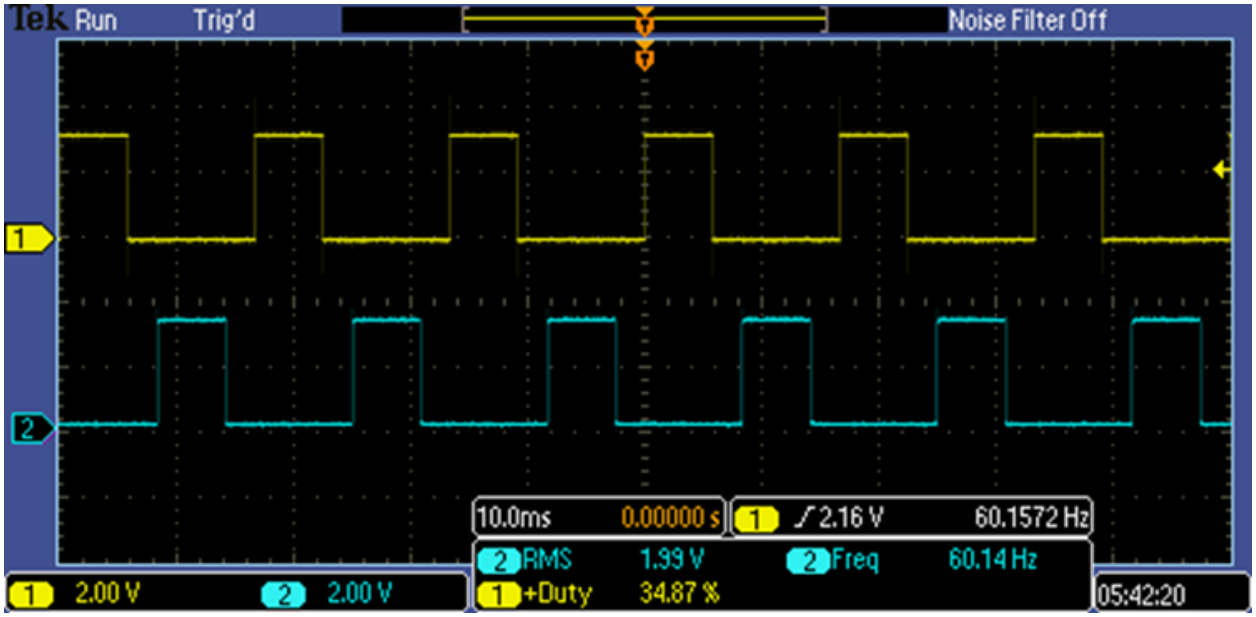


Figure 4: Control Signals from pins J6.5 and J2.8 of the Launchpad Microcontroller

Step 4: Modified Sine-Wave Inverter Testing

The Input and Output of the Modified Sine-Wave Inverter were tested with a 250 W load. The results are shown below. The Duty Cycle shown is the Duty Cycle of the control signal- it goes high for 34.87% of the cycle, and low for 34.87% of the cycle.

Frequency: 60.14 Hz

Control Signal Duty Cycle: 34.87%

Input HVDC Voltage: 149.6 V

Output RMS Voltage: 119 Vrms

Output RMS Current: 0.4915 Arms

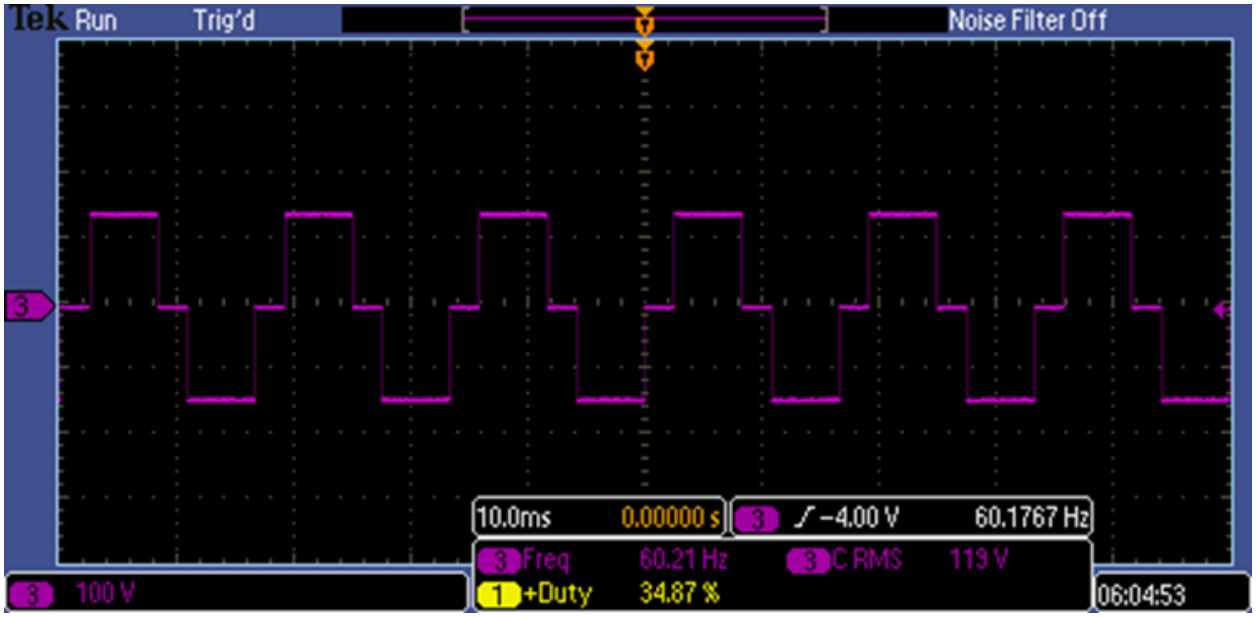


Figure 5: Oscilloscope measurement of the Output AC Voltage of the Modified Sine-Wave Inverter

Step 5: True Sine-Wave Inverter

A 1 microFarad Capacitor will be used for the output filter. The bandwidth of the filter should be much greater than 60 Hz, and much smaller than our switching frequency of 20 kHz. A bandwidth of 3200 Hz was selected. This yields an inductance of

$$\frac{1}{(2\pi f)^2} * \frac{1}{2C} = \frac{1}{(2\pi * 3200Hz)^2} * \frac{1}{2 * 1microFarad} = 1.24\,mH$$

Assuming the following values for the inductor design, estimating a maximum inductor resistance of 250 mOhms,

Bmax = 0.5 T
Rho = $1.724*10^{-6}$ Ohm*cm
L = 1.3 mH
Imax = 0.8 A
Ku = 0.5
R = 250 mOhms

As PQ32/20 cores are the only ones available, those cores will be used. They have the following measurements:
Kg = 0.203 cm^5
Ac = 1.70 cm^2
Wa = 0.471 cm^2
MLT = 6.71 cm
Lm = 5.55 cm

The requirements of the core geometric constant can be calculated as:

$$Kg < \left(L * \frac{Imax}{Bmax}\right)^2 * \frac{Rho}{R * Ku} * 10^8 cm^5 = 0.0060\ cm^5$$

With a geometric constant of 0.203 cm^5 the PQ32/20 core will meet this requirement.

The number of turns can be calculated as:

$$n = \frac{LImax}{BmaxAc} * 10^4 = 17.4$$

This number will be rounded up to 18, giving 18 turns.

The air-gap length can be calculated as:

$$Lg = \mu_0 * Ac * \frac{n^2}{L} * 10^{-4} = 2.71 * 10^{-5} m = 0.0011\ inches$$

Giving an air-gap length of 0.0011 inches.

The wire gauge selected can be calculated as:

$$Aw < Ku * \frac{Wa}{n} = 18.1 * 10^{-3} cm^2$$

AWG #18 wire gauge fits this requirement.

Checking that the Resistance meets initial estimations,

$$R = Rho * n * \frac{MLT}{Aw} = 18.2\ mOhms$$

It does.

Now checking that the constraints of the Core Geometric Constant are met with the calculated resistance,

$$Kg < \left(L * \frac{Imax}{Bmax}\right)^2 * \frac{Rho}{R * Ku} * 10^8 cm^5 = 0.082\ cm^5$$

It does.

The inductor measurements are shown below.

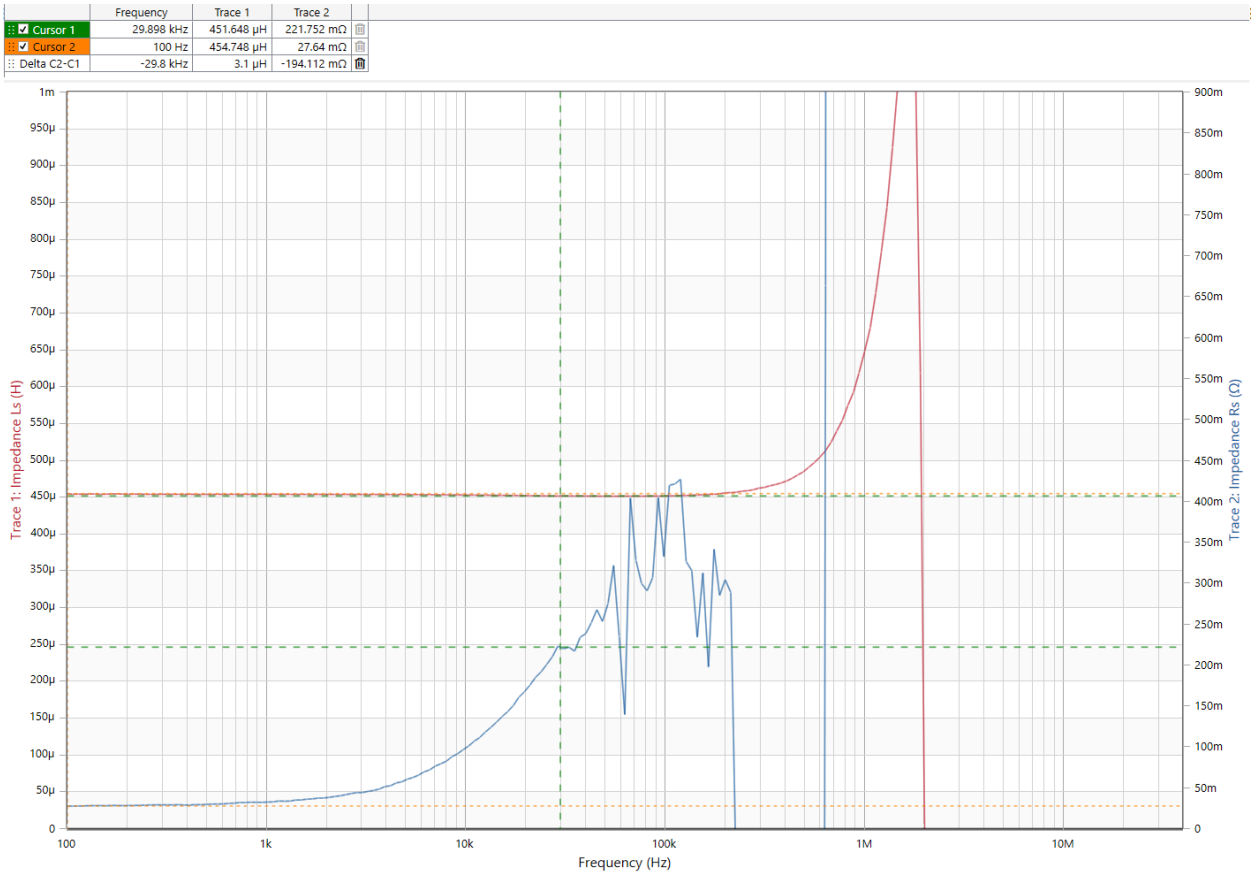


Figure 6: L1 bode 100 network analyzer plot

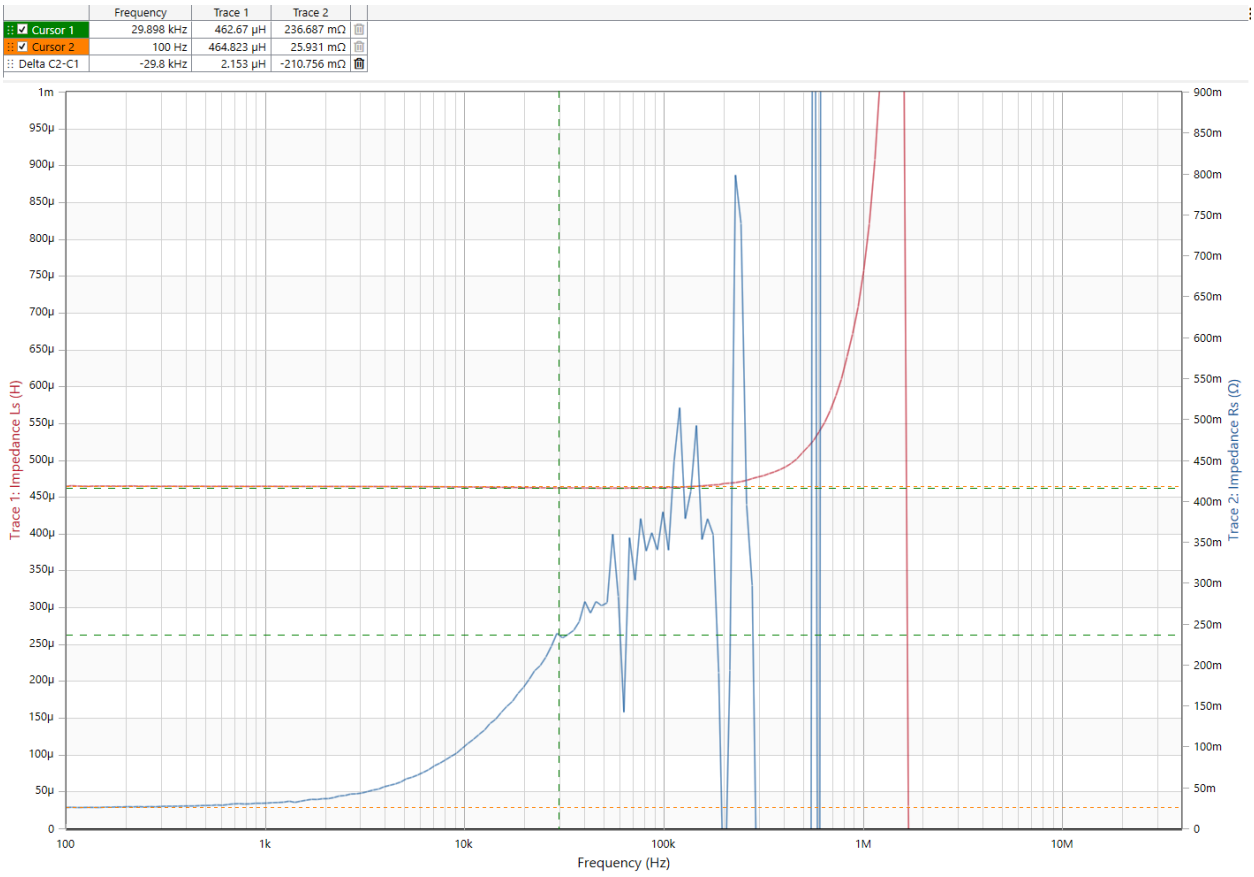


Figure 7: L2 Bode 100 network analyzer plot

The measured inductances and series resistances of our inductors are:

Inductor	Inductance	Series Resistance
L1	452 uH	27.6 mOhm
L2	463 uH	25.9 mOhm

The code of PWM_ADC3.c was updated to achieve a true sine wave as follows.

```
14 void InitADC_sampling(void);                          // ADC input on J5.3 of the
15 void update_duty(unsigned int);                       // Update PWM duty cycle
16 interrupt void adc_isr(void);                         // ADC interrupt service rou
17
18 unsigned int sensed_voltage,sensed_current;           // 12-bit ADC output (0-to-4
19 unsigned int pulse_width;                             // PWM pulse width expressed
20 unsigned int period;                                  // PWM period expressed as a
21 unsigned int half_duty;                               // PWM pulse width expressed
22 unsigned int mod_period;                              // PWM period expressed as a
23 unsigned int sin_clock; // Tracks when to update duty cycle with new sine value
24 unsigned int sin_index; // Tracks index of sin_vals that is to be used to update
25 unsigned int num_vals=199;
26 float sin_vals[199] = {300.0000000000, 309.4705649294, 318.9316893938, 328.373942338
27
28
29 // These are defined by the linker (see F28027.cmd)
```

```
86    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;
87    EDIS;
88
89    period = 500;       // PWM period for EPWM1 set as the number of (1/60 MHz) clock
90    mod_period = 600; // PWM period for EPWM3 and EPWM 4 set as the number of (1/7.5
91    half_duty  = 300; // PWM duty on EPWM3 AND EPWM4 to setup 120 Hz square wave sign
92    sin_clock = 0;
93    sin_index = 0;
94
95    InitEPwm1Gpio();                    // Sets up the GPIO as PWM pins
96    InitEPwm3Gpio();                    // Sets up the GPIO as PWM pins
97    InitEPwm4Gpio();                    // Sets up the GPIO as PWM pins
98    InitEPwm1Example(period, 0);        // PWM setup with pulse width set to zero
99    InitEpwm3_4(mod_period, half_duty); // PWM setup with pulse width set to 50% duty
100   InitADC_sampling(); // ADC setup
```

```
// Update duty cycle by updating the pulse_width for the PWM output on J6.1 of the F
void update_duty(unsigned int pulse_width)
{
    EALLOW;
    EPwm1Regs.CMPA.half.CMPA = pulse_width;
    EDIS;
}

// Update duty cycle by updating the pulse_width for the PWM output on J6.1 of the F
void update_sin_duty(unsigned int pulse_width)
{
    EALLOW;
    EPwm3Regs.CMPA.half.CMPA = pulse_width;
    EPwm3Regs.CMPB = pulse_width;
    EDIS;
}

// ADC interrupt service routine, update sensed_voltage
interrupt void  adc_isr(void)
{
  sensed_voltage = AdcResult.ADCRESULT0;    // Voltage sensed on pin J5.3 of the F28
  sensed_current = AdcResult.ADCRESULT1;    // Voltage sensed on pin J5.4 of the F28

  update_duty(pulse_width);    //Changes the duty cycle of the PWM output signal on

  if (sin_clock >=9) { // Update duty after 9 interrupts (100 kHz interrupt freq,
                       // This gives about 60 Hz sin wave using 199 points in the
                       // sine wave.)
      update_sin_duty(sin_vals[sin_index]);
      if (sin_index>=num_vals) { // Wrap index to beginning of sin_vals
          sin_index = 0;
      } else {
          sin_index += 1;
      }
      sin_clock = 0;
  } else {
      sin_clock += 1;
  }

  EALLOW;
```

```
void InitEpwm3_4(unsigned int mod_period, unsigned int half_duty)
{
    // EPWM 3 SETUP
    // Setup TBCLK
    EPwm3Regs.TBPRD = mod_period;                // TBPRD = (TIME_BASE_CLOCK_FREQ/PW
    EPwm3Regs.TBPHS.half.TBPHS = 0x0000;         // Phase is 0. This register deterr
    EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;  // Sync down-stream module
    EPwm3Regs.TBCTR = 0x0000;                    // Clear counter for trailing-edge

    // Set Compare values
    EPwm3Regs.CMPA.half.CMPA = half_duty;            // This sets the initial duty
    EPwm3Regs.CMPB = half_duty;                      // This sets the initial duty

    // Setup counter mode
    EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;   // This sets the PWM type, ie trail
    EPwm3Regs.TBCTL.bit.PHSEN = TB_ENABLE;       // Enable phase loading
    EPwm3Regs.TBCTL.bit.HSPCLKDIV = 0;           // Clock ratio to SYSCLKOUT
    EPwm3Regs.TBCTL.bit.CLKDIV = 1;              // PWM clock prescale set to 8. Tir

    // Setup shadowing
    EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // Load on Zero
    EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;

    // Set actions
    EPwm3Regs.AQCTLA.bit.ZRO = AQ_SET;           // Sets PWM1A J6.5 when period star
    EPwm3Regs.AQCTLA.bit.CAU = AQ_CLEAR;         // Clears PWM1A J6.5 when CMPA > sa

    EPwm3Regs.AQCTLB.bit.ZRO = AQ_CLEAR;           // Sets PWM1B J6.6 when period st
    EPwm3Regs.AQCTLB.bit.CBU = AQ_SET;         // Clears PWM1B J6.6 when CMPB > sawt

//    // Set Deadband
```

Pins J6.5 and J6.6 were made as complementary PWM signals and used to control each side of the half-bridge inverter. A set of 199 pulse_width values corresponding to one sinusoidal period are iterated through using the adc_isr. The frequency of the adc_isr is about 100 kHz, and we found that updating the duty cycle once every 9 interrupts, with a set of 199 values in a single sinusoidal period, resulted in a 60.19 Hz sine wave. The frequency of the PWM signals was set to 50 kHz, which is one decade above the pole frequency of our filter, 5.26 kHz, in order to properly attenuate switching noise.

The inverter was run with a 250 Ohm load, and input voltage increased to achieve 120 V RMS output. The following operating conditions were measured.

Input Voltage:	186.7 V
Output Voltage (RMS):	120.1 V
Output Current (RMS):	0.58 A
Voltage Ripple:	2.60 V

The output waveform and voltage ripple can be seen in the figures below.

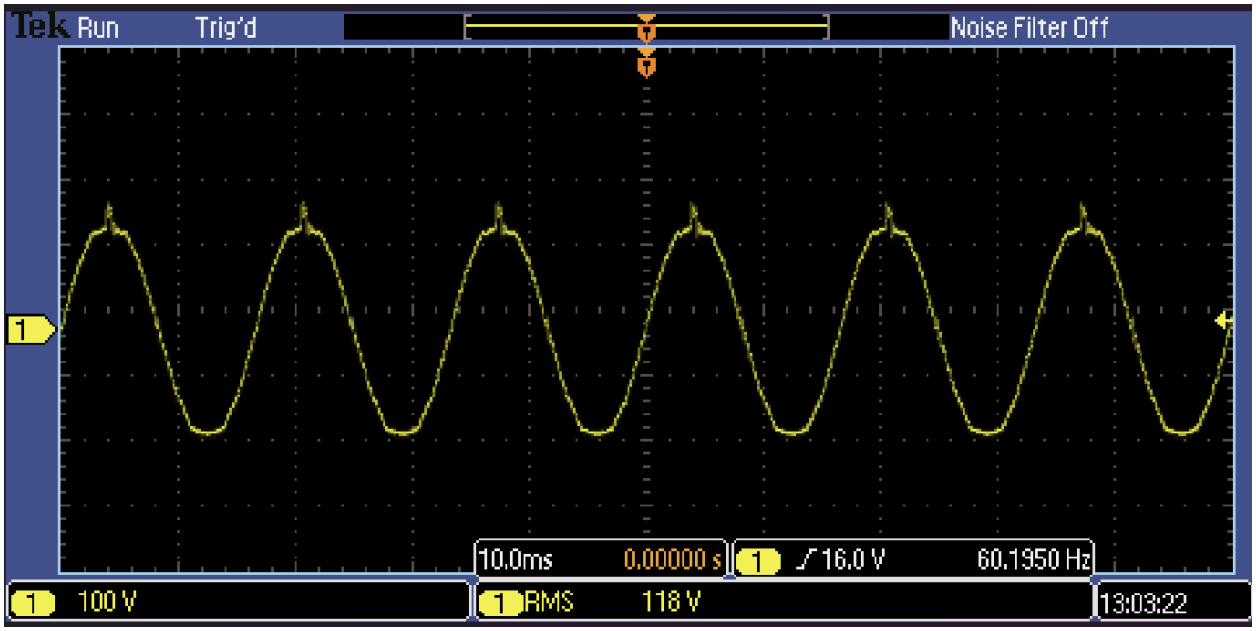


Figure 8: Output voltage at nominal operating point of 120 V rms out

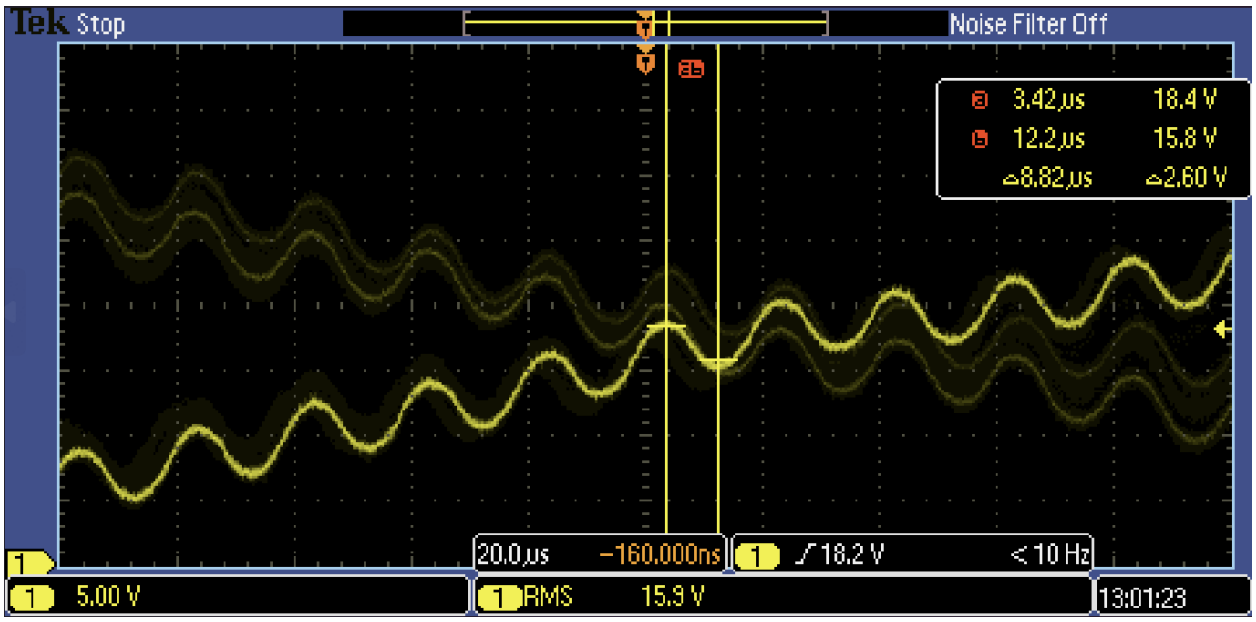


Figure 9: Voltage ripple at nominal operating point

We were successfully able to produce a true sine wave with our inverter, at 60Hz and 120 V RMS output, with minimal voltage ripple.