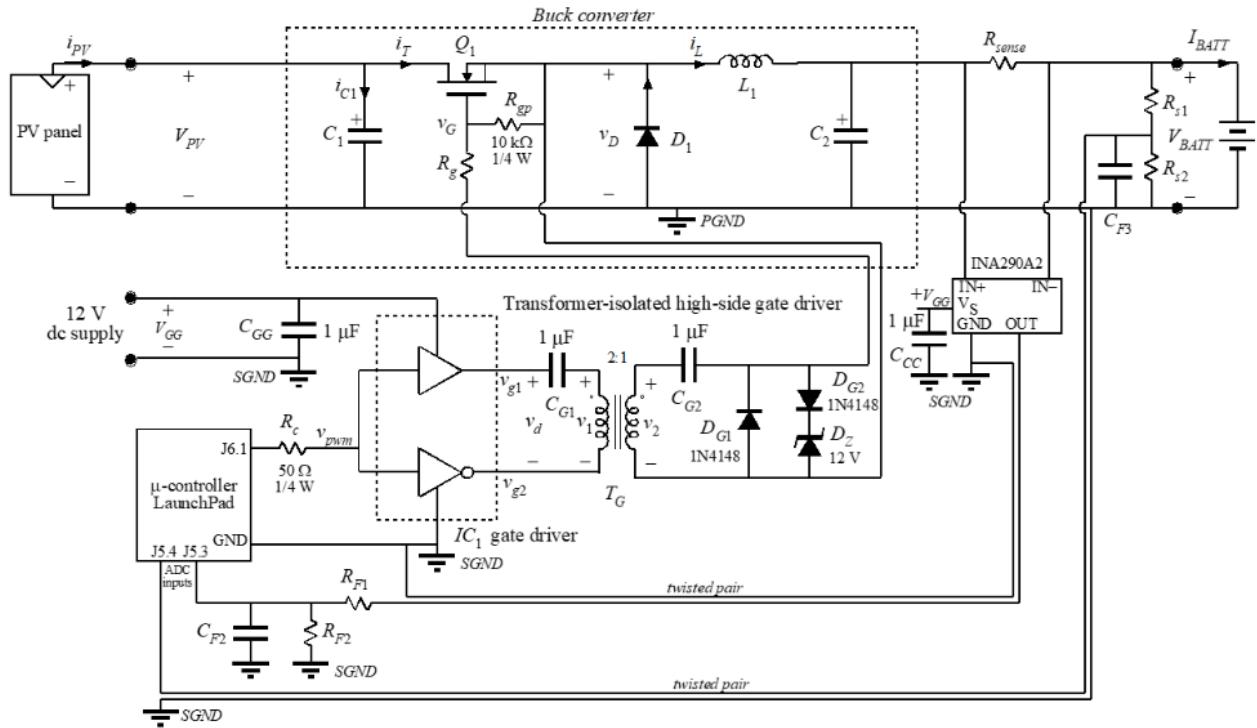


Experiment #3 Part 2  
PV Power Electronics Laboratory

Completed By:  
Zach Shelton & Shane McCammon

## Step 1:



$$R_{S1} = 47\text{k}\Omega$$

$$R_{S2} = 10\text{k}\Omega$$

$$C_{F3} = 100\text{nF}$$

$$R_{F1} = 22\text{k}\Omega$$

$$R_{F2} = 10\text{k}\Omega$$

$$C_{F2} = 100\text{nF}$$

$R_{S1}$  and  $R_{S2}$  were chosen to limit the voltage going to the J5.4 pin on the Launchpad. This is more than enough since the max voltage needed to reach 3.3 volts at the pin would be 18.81 V.

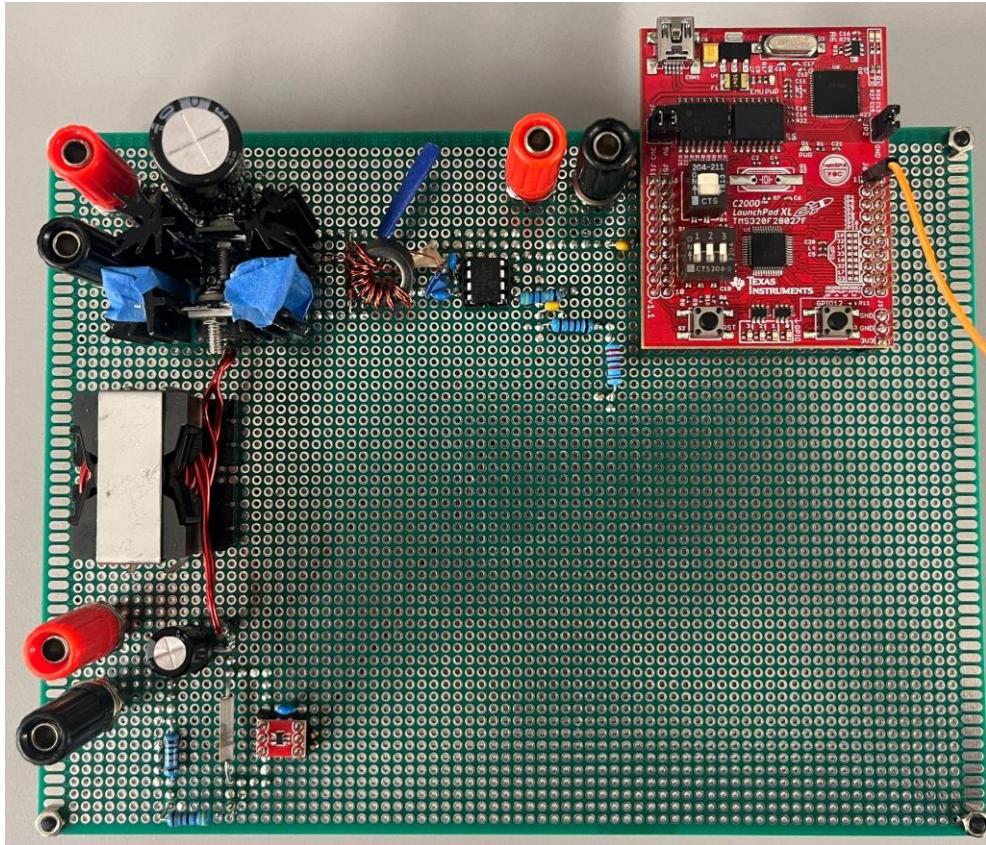
$(18.81 * (10\text{k}) / (10\text{k} + 47\text{k})) = 3.3$ .  $R_{F1}$  and  $R_{F2}$  were chosen similarly. The capacitor values were picked as 100nF to filter high frequency noise.

$$\text{sensed\_voltage} = V_{\text{batt}} * 4095 / 3.3 * (10\text{k} / 10\text{k} + 47\text{k})$$

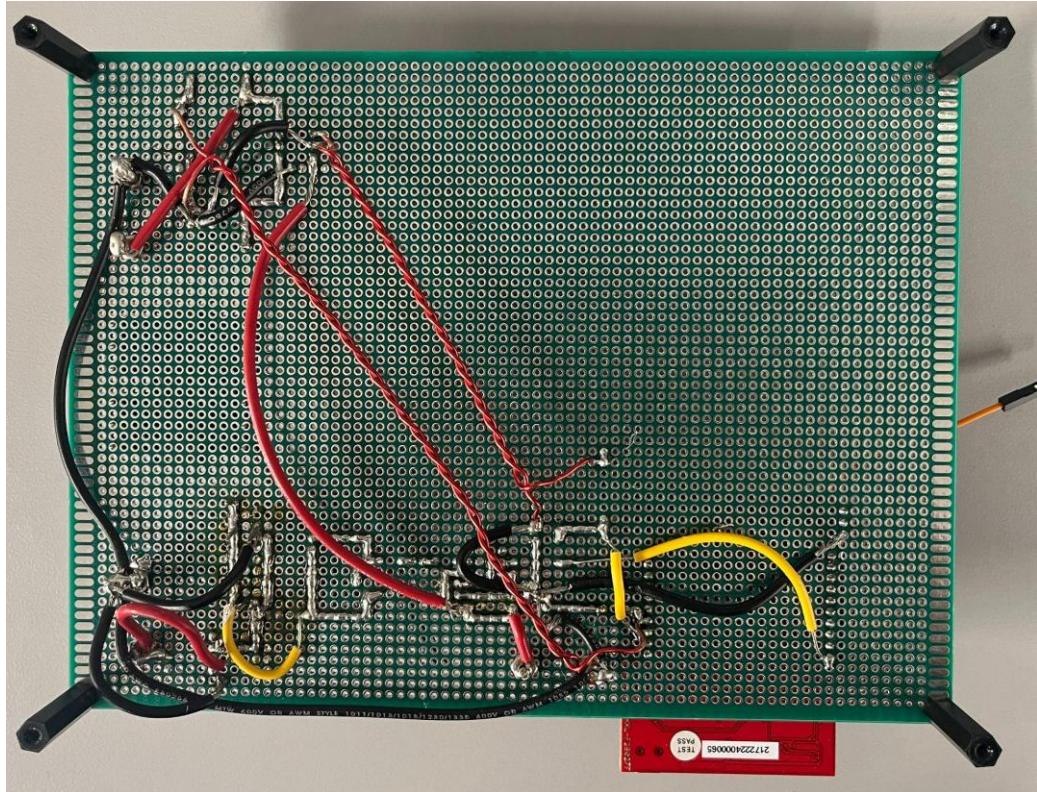
$$\text{sensed\_current} = I_{\text{batt}} * 50 * R_{\text{sense}} * 4095 / 3.3 * (10\text{k} / 10\text{k} + 22\text{k})$$

## Step 2:

The top and bottom view of the circuit are shown below:



**Top View of Buck Converter Circuit with Feedback**



**Bottom View of Buck Converter Circuit with Feedback**

To start with testing, we checked for short circuits across all power rails and checked all conductivity of all newly added components to the rest of the circuit before turning on power. Next, we checked the voltages at each node and ensured those were correct. Finally, we checked that the sensed\_voltage and sensed\_current in the code matched actual measured values. Anytime we found issues, we probed the circuit to find where we had a problem and then fixed it. Some errors were very hard to detect and we had a few components go up in smoke. After making adjustments, we were able to move on.

### Measured Results Compared to Sensed Results

Measured V <sub>batt</sub> [V]	Measured I <sub>batt</sub> [A]	sensed_voltage	sensed_current
12	4	2612.44 = 12V	2256.20 = 4A
13	4.333	2830.14 = 12.9V	2442.33 = 4.334A
14	4.667	3043.01 = 13.98V	2630.22 = 4.663A
15	5	3270.59 = 15.02V	2820.25 = 5A

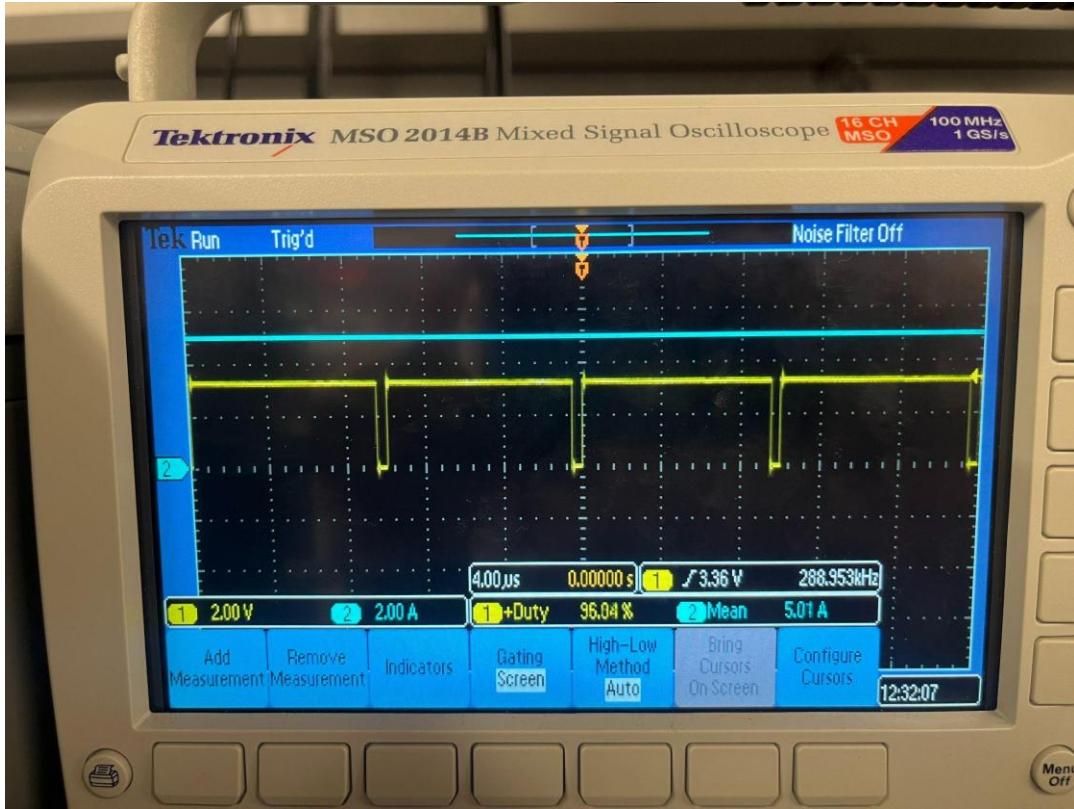
The results from this test shows that the sensed\_voltage and sensed\_current measurements in code are accurate and we are able to proceed with the rest of the lab.

### Step 3:

We used a perturb and observe method for the maximum power point tracking algorithm. This algorithm increases the duty cycle as the power increases until the observed power begins to decrease meaning it has reached a peak. We average the power for 10000 measurements due to possible fluctuations in sensed values. The overcharge protection function just checks when the sensed\_voltage is too high and then it stops current from going through the circuit for 30 seconds allowing the battery to discharge. This is checked every time before changing the duty cycle to ensure there is no overcharging. It keeps the battery voltage within a range of around 12.8 to 13.4 volts at a maximum. This can be easily changed through adjusting the code. The code is on the last page of the report.

For testing, we chose a  $V_{in} = 30.36V$  and  $R_{in} = 2.7\Omega$ . We ran the code for around 2 minutes to allow it to find the maximum power point and see if it would stay there. Initially, the duty cycle was skyrocketing to the maximum value. This was due to an error in the conditional statements where it was never checking if the power was decreasing. After fixing this, the code was functional with the circuitry and we were able to move on.

Below is a scope capture of the converter operating at maximum power point:



**Scope Capture at Maximum Power Point**

We found that the maximum power point was around 95% duty cycle.

To make sure the maximum power point was correct, we manually tested different pulse\_width values until the maximum power point was found there. With the setup we had, we found the maximum power point to be about 95% which was consistent with what the algorithm found as well so we knew it was working.

Converter Efficiency at the Maximum Power Point:

$$V_{in} = 14.45V$$

$$I_{in} = 5A$$

$$P_{in} = 72.25W$$

$$V_{out} = 13.9V$$

$$I_{out} = 5A$$

$$P_{out} = 69.5W$$

$$\eta_{mpp} = 96.19\%$$

*It should be noted that the overcharge protection was not implemented when this data was collected.*

To ensure that the charge control feature was working properly, we let the program and the circuit run for around 20 minutes and monitored the battery voltage to make sure it stayed within a reasonable voltage. The voltage never exceeded 13.4V at the maximum point and stayed between that and roughly 12.8V. If we decide this is too high, it is very simple to edit the code by changing what value the feature is enabled at in the conditional statement.

### Step 3 Observe and Perturb Algorithm Code:

```

132 // ADC interrupt service routine, update sensed_voltage
133 interrupt void adc_isr(void)
134 {
135
136     if(counter > 60000)
137     {
138
139         for(i = 0; i < 10000; i++)
140         {
141             sensed_voltage = AdcResult.ADCRESULT1; // Voltage sensed on pin J5.3 of the F28027 LaunchPad, 0-to-(2^12 -1) corresponds to 0-to-3.3 V
142             sensed_current = AdcResult.ADCRESULT0; // Voltage sensed on pin J5.4 of the F28027 LaunchPad, 0-to-(2^12 -1) corresponds to 0-to-3.3 V
143             current = ((float)(sensed_current)/387.7840877); // Constant given by 4095/(3.3*3.2) (3.3 is the max voltage at the pin and 3.2 is the voltage divider ratio)
144             voltage = ((float)(sensed_voltage)/217.7033493); // Constant given by 4095/(3.3*5.7) (3.3 is the max voltage at the pin and 5.7 is the voltage divider ratio)
145             loop_power = voltage * current;
146             power_sum += loop_power;
147         }
148
149         power = power_sum/10000;
150
151         power_sum = 0;
152
153         // Over-charge protection
154         if(voltage > 14)
155         {
156             update_duty(10);
157             DELAY_US(30000000); //set to 1.6% for 30 seconds to discharge the battery
158         }
159
160         else
161         {
162
163             // *****MPP Tracking*****
164             //Check if the power is increasing and increase PWM towards the peak
165             if(power > last_power)
166             {
167                 pulse_width += (increment*direction);
168             }
169
170             // Check if we've passed the peak value and re-adjust back down to it
171             else if(power < last_power)
172             {
173                 direction = -direction;
174                 pulse_width += (increment*direction);
175             }
176
177             // Define a maximum PWM to avoid crashing
178             if(pulse_width > 588)
179             {
180                 pulse_width = 588;
181             }
182
183             // Define a minimum PWM to avoid crashing
184             else if(pulse_width < 10)
185             {
186                 pulse_width = 10;
187             }
188
189             // Update the duty cycle accordingly
190             update_duty(pulse_width);
191
192             // Set the last power for new comparisons
193             last_power = power;
194         }
195
196         counter = 0;
197     }
198
199     counter++;
200
201 //update_duty(pulse_width); //Changes the duty cycle of the PWM output signal on pin J6.1
202
203 EALLOW;
204 EPwm2Regs.CMPA.half.CMPA = (period)>>1; // Starts the sampling of the current and voltage signals at 0.5*(D.Ts) for every succeeding switching cycle to sample average values and avoid ringing.
205 EDIS; // This could be changed to whatever the student needs based on duration of ringing in their system
206
207 AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1; // Clear ADCINT1 flag reinitialize for next SOC
208 PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Acknowledge interrupt to PIE
209
210 return;
211 }
```

## Step 4 Complete System Test:

March 9, 2025 / 12:08 / Irradiance: 1028 W/m<sup>2</sup>

Converter Efficiency at the Maximum Power Point (unshaded):

$$V_{PV} = 14.8V$$

$$I_{PV} = 4.8A$$

$$P_{in} = 71W$$

$$V_{out} = 12.9V$$

$$I_{out} = 4.7A$$

$$P_{out} = 61W$$

$$\eta_{mpp} = 86\%$$

Using MPPT we can currently maximize our power transfer to 61W with an output current of 4.7A. If we did not use MPPT and just relied on direct energy transfer the PV panel would be forced to match the battery voltage which would significantly reduce our output power. If we can improve our losses we should be able to transfer around 78W with our calculations from experiment 2 and charge the battery much faster than if we were to use direct energy transfer methods. We continue to have issues with minor shorts and excessive noise in our system that we are continuously addressing and trying to prevent.

Converter Efficiency at the Maximum Power Point (shaded):

$$V_{PV} = 14.2V$$

$$I_{PV} = 2A$$

$$P_{in} = 28.4W$$

$$V_{out} = 12.9V$$

$$I_{out} = 1.6A$$

$$P_{out} = 21W$$

$$\eta_{mpp} = 74\%$$

Shading the panel created a significant drop in current seen from the panel. There was also a decrease in efficiency, as we have seen our converter become much less efficient at lower power levels. These results align with our expected behavior for a partially shaded panel although it seems some light may have been illuminating the “shaded panels” as the panel voltage still remained around 14V and it was expected it to drop more than that. The shading was done by a piece of paper over the cells. MPPT results still yield better power transfer than direct energy transfer as the system is constantly tracking the optimal duty cycle for performance.