Experiment #3
DC-DC Converter for Maximum Power Point Tracking (MPPT) and Charge Control

Part 2:
Maximum Power Point Tracking and Charge Control

Andrew Thibeault and Dane Thorn
ECEN 5517
2023-02-27

Introduction:

The objective of Part 2 of this experiment is to demonstrate a closed loop control scheme for the buck converter developed in Part 1 that implements maximum power point tracking and battery charge control. Voltage and current sensing circuitry is designed to mitigate noise and disturbances in the sensed signals, and these signals are A/D converted for processing and control by the LaunchPad Microcontroller.

Step 1: Battery current and voltage sensing

A schematic of the sensing circuitry for the output current and output voltage of the buck converter is shown below.

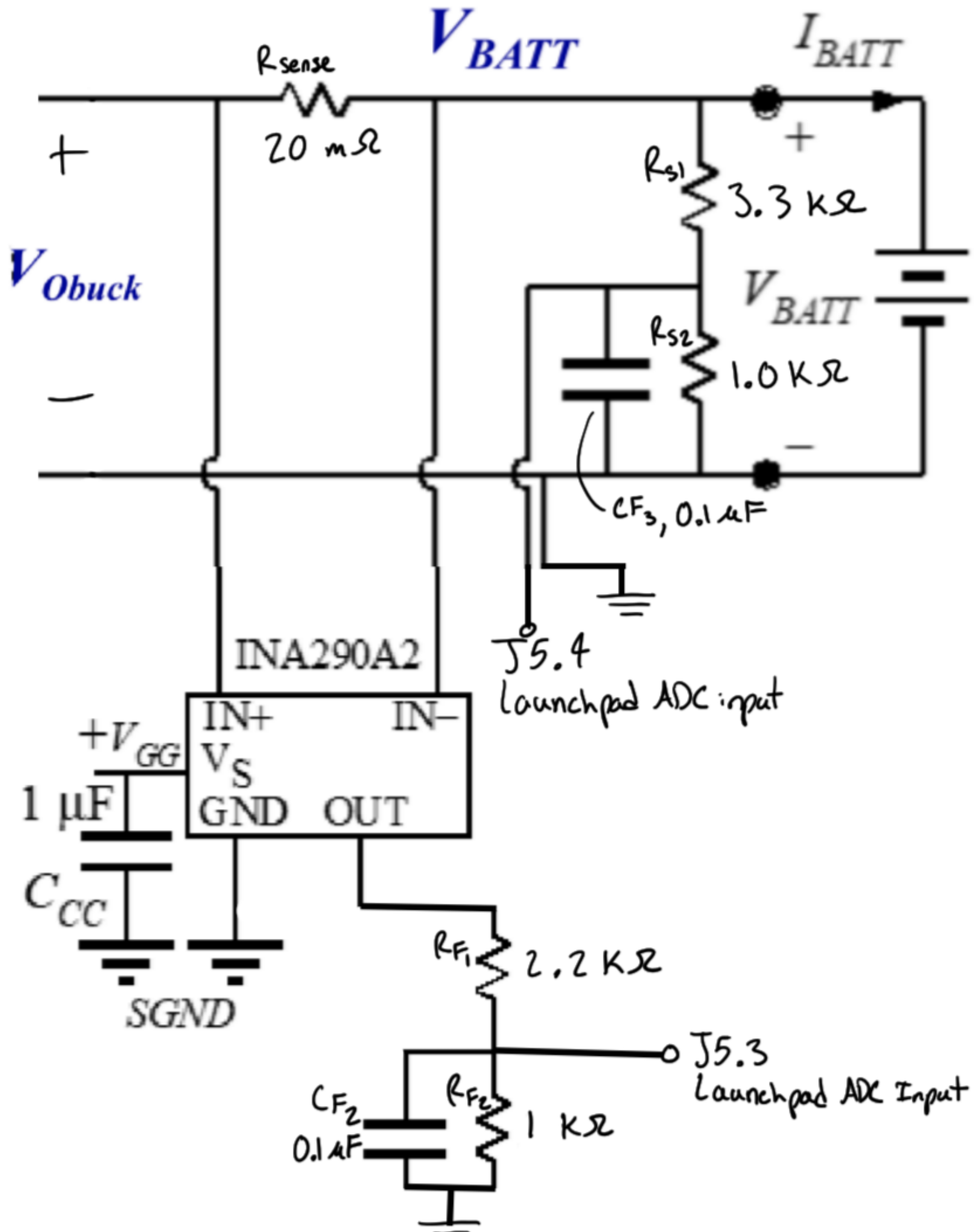


Figure 1: Buck Converter Sensing Schematic

Voltage dividers are used to step down the voltage input to the ADC to a maximum of 3.3 V, the maximum voltage that the ADC can convert. For the voltage sensor, assuming a maximum V_{Batt} voltage of 14 V, a 4.3:1 voltage divider is used with $R_{S1} = 3.3\ k\Omega$ and $R_{S2} = 1.0\ k\Omega$. This maximum V_{Batt} comes from the assumption that the battery voltage will not rise above about 13.2 V, with some extra range for safety. For the current sensor, assuming a maximum voltage at INA290A2_OUT of 10 V, a 3.2:1 voltage divider is used with $R_{F1} = 2.2\ k\Omega$ and $R_{F2} = 1.0\ k\Omega$. This maximum voltage limit comes from the assumption that the maximum battery current will be below 10 A, and $V_{INA290A2\_OUT} = 50*R_{Sense}*I_{Batt} = I_{Batt}$.

A lowpass filter is implemented in each sensing circuit in order to filter out high frequency switching harmonics and noise, but to still allow circuit dynamics through. A cutoff frequency of a few kHz is used in each case, and a 0.1 μF capacitor is used for both circuits. This gives a $f_{Cutoff\_VoltageSensing} = 2.07$ kHz and a $f_{Cutoff\_CurrentSensing} = 2.31$ kHz.

The resultant expressions for ADC input voltages from the voltage and current sensing circuits are:

$$V_{CurrentSensing} = 50 * R_{Sense} * I_{Batt} * R_{F2} / (R_{F1} + R_{F2}) = 0.313 * I_{Batt}$$

and

$$V_{VoltageSensing} = V_{Batt} * R_{S2} / (R_{S1} + R_{S2}) = 0.233 * V_{Batt}$$

Step 2: Testing of battery current and voltage sensing

The voltage and current sensing circuits were integrated into the converter as shown below.

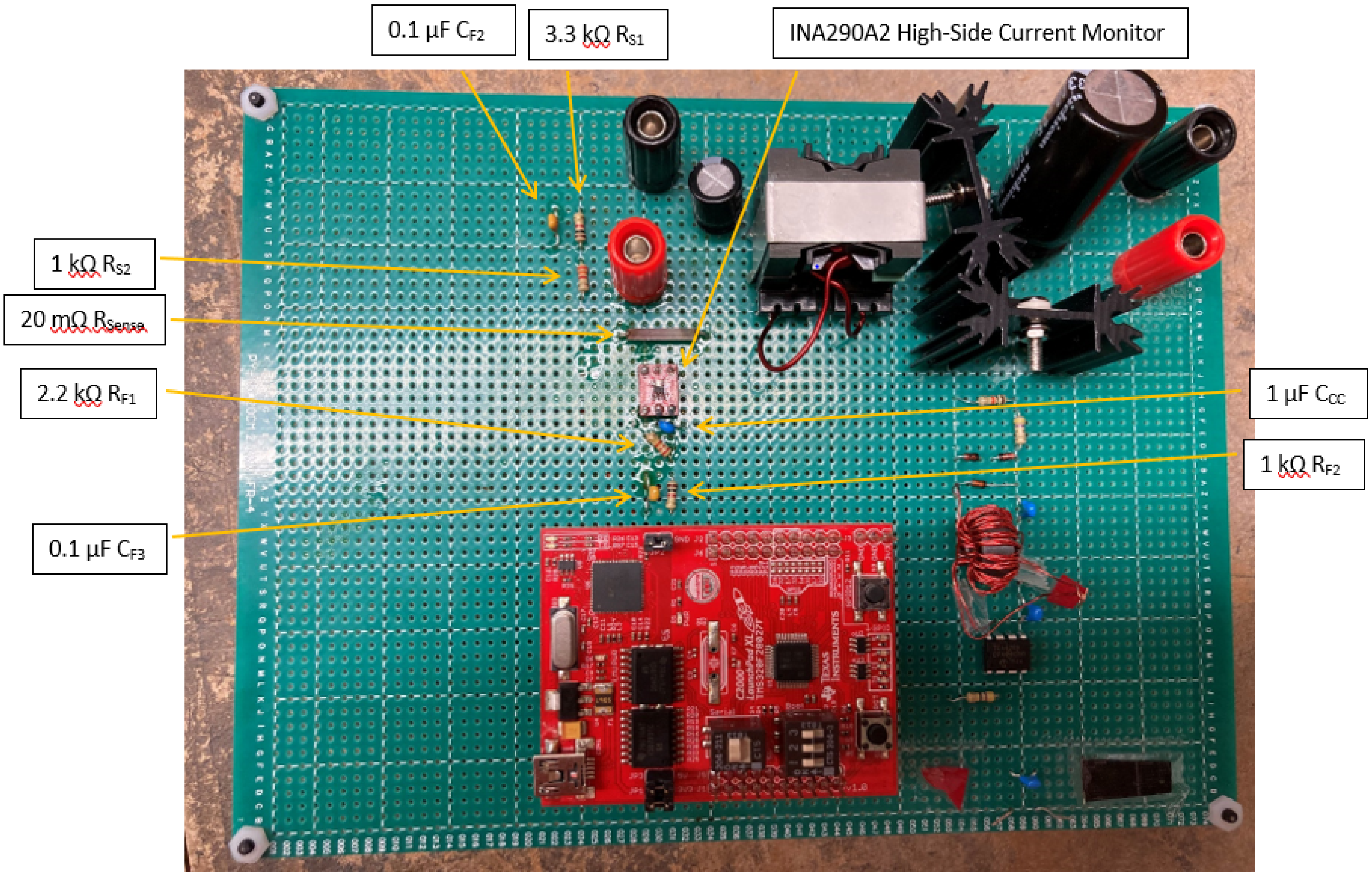


Figure 2: Implemented Voltage and Current Sensors with the resistors, capacitors, and high-side monitor indicated

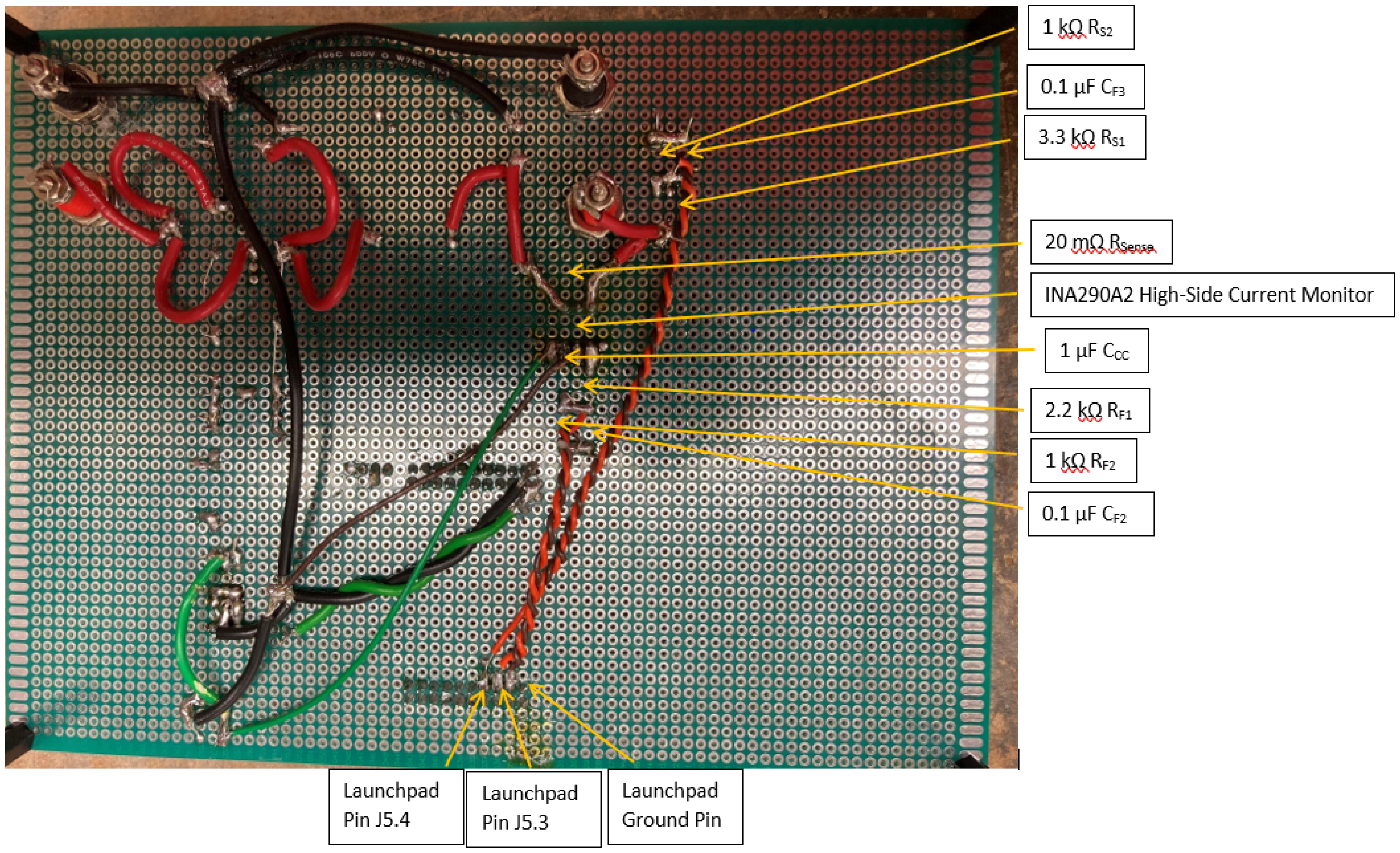


Figure 3: Wiring of sensing circuit with circuit components indicated. The arrows are pointed at where the middle of each component is on the top side of the breadboard

Debugging:
When testing our sensed voltage and current, we noticed that our sensed voltage was what we would expect from our sensed current, and vice versa. Changing which pin the sensed current and sensed voltage were taken from in our code fixed this issue.

Testing:
To test the operation of our voltage and current sensing circuitry we ran the PWM_ADC code with a pulse_width of 495, giving a duty cycle of 0.825. The input voltage of the Buck Converter was set to 17.2 Volts using a DC Power Supply. The Buck Converter output load was a Rheostat, and the output voltage and current were monitored using two multimeters. The output voltage and current, as well as the voltage and current sensed by Code Composer Studio, are discussed below.

Voltage Sensing:
Voltage at Converter Output:
13.9 V

Sensed Voltage:
4069 Bits
Converting our sensed voltage reading from Bits to Volts, we find:
Sensed_voltage * (Normalized Pin Voltage / Normalized Pin Bits) * ((R_{S1} + R_{S2}) / R_{S2}) =
4069 Bits * (3.3 V / 4095 Bits) * (4.3 Ohms / 1 Ohm) = 14.1 Volts

14.1 Volts / 13.9 V = 1.036 = 101.4%
This calculation shows our voltage sensing circuit operates with an accuracy of +1.4%, which we deemed sufficient for our purposes.
Current Sensing:
Current at Converter Output:
4.81 A

Sensed Current: 1845 Bits

Converting our sensed current reading from Bits to Amps, we find:
Sensed_current * (Normalized Pin Voltage / Normalized Pin Bits) * ((R_{F1} + R_{F2}) / R_{F2})
* (1 / (50*R_{sense})) =
1845 Bits * (3.3 V / 4095 Bits) * (3.2 Ohms / 1 Ohm) * (1 / (50 * 0.02 Ohms)) = 4.76 Amps

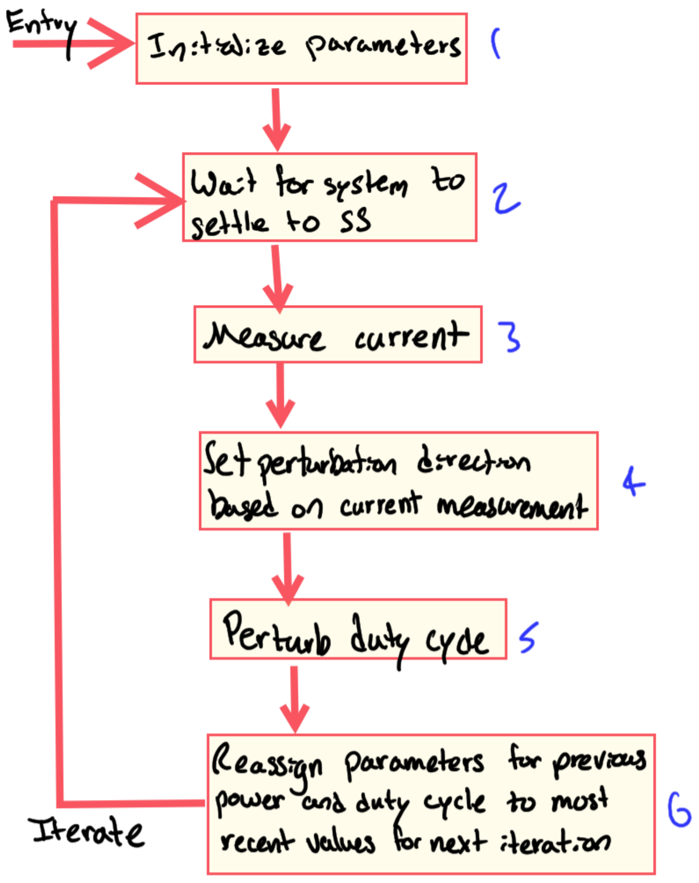
4.76 A / 4.81 A = 0.973 = 98.9 %
This measurement shows our current sensing circuity operating with an accuracy of -1.1%.

As both our sensed voltage and current were within a few percent of the values measured on the multimeters, we felt our sensing circuity was accurate enough that we were confident in proceeding.

Step 3: MPPT and charge-control development and bench testing

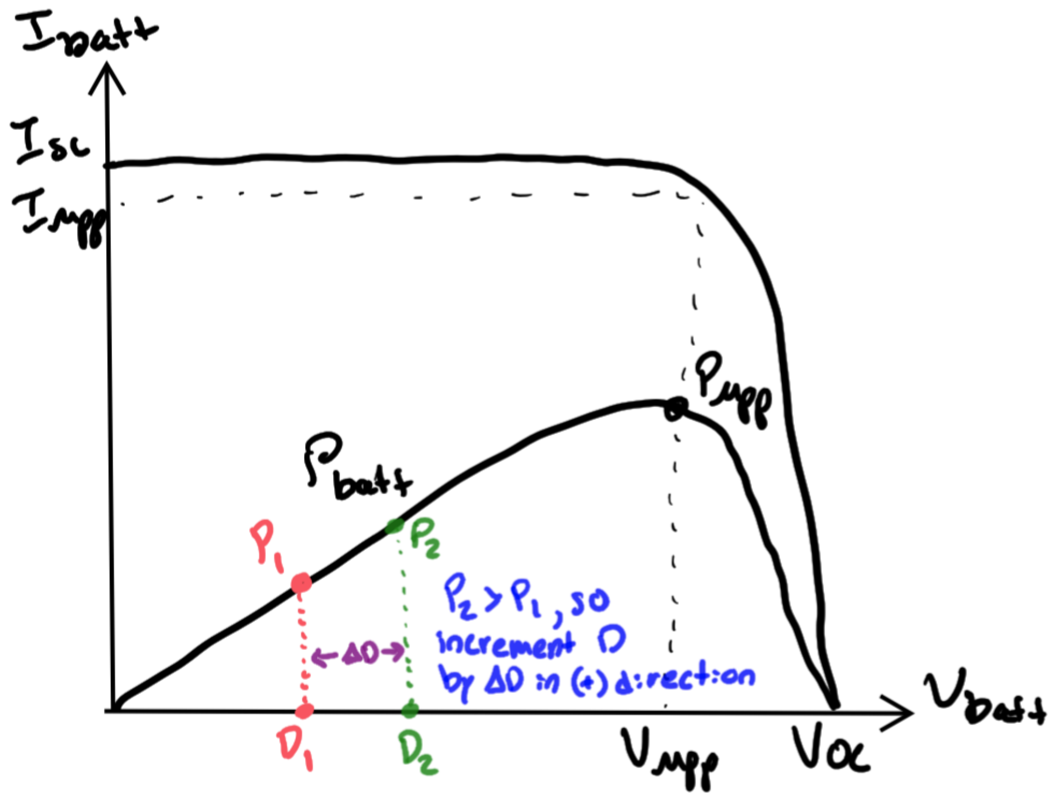
A perturb-and-observe style maximum power point tracking (MPPT) algorithm was implemented on the launchpad.

In this algorithm, the duty cycle of the converter is iteratively incremented in the direction that produces the greatest current input to the battery. After an initial duty cycle is set, the algorithm waits 20 ms for the system to settle. The battery power is observed at this duty cycle setting. If the current is greater than the last observation, the duty cycle is perturbed by a fixed increment in the direction that produced the greater current. If the current is less than the last observation, the direction of perturbation is opposite. Repeat this process from the wait-to-settle point (2 in figure 4). The duty cycle should approximately converge to the value that results in maximum current to the battery. A pulse_width increment size of 15 was found to track the maximum power point well, when combined with the 20 ms wait-to-settle time.



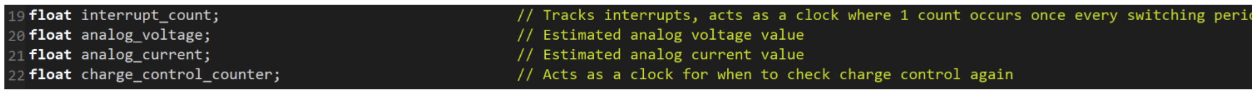
Entry
Initialize parameters
1
Wait for system to settle to SS
2
Measure current
3
Set perturbation direction based on current measurement
4
Perturb duty cycle
5
Reassign parameters for previous power and duty cycle to most recent values for next iteration
6
Iterate

Figure 4: Perturb and Observe Algorithm Flowchart

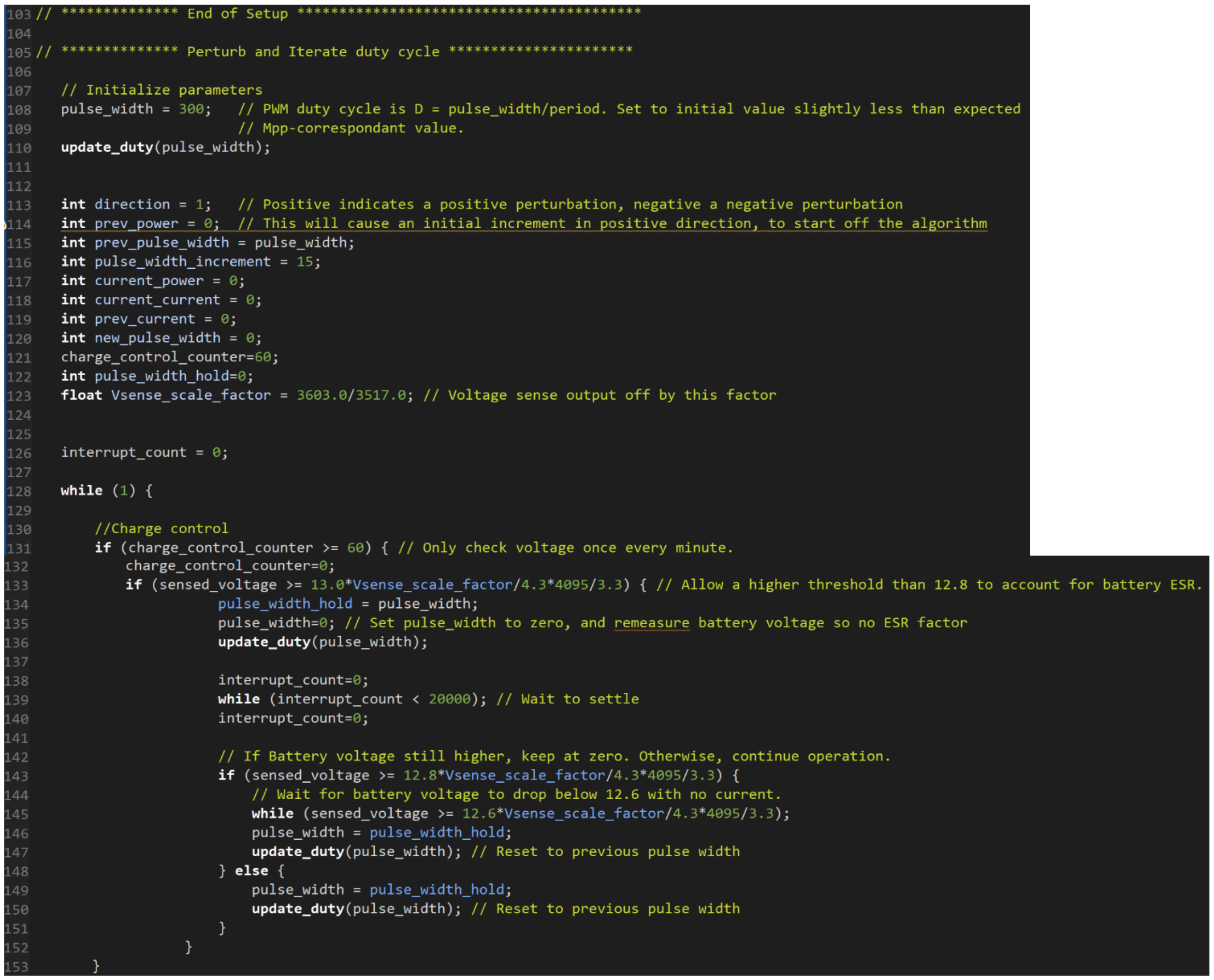


A charge-control feature was implemented to turn off the buck converter when the battery voltage rises above 12.8 V. This feature runs once every minute, checking if the sensed voltage is at or above 13.1 V. The reason for this higher voltage is that a voltage drop across the battery equivalent series resistance will be present when current is flowing in the battery. This 13.1 value is a conservative estimate of what we can expect to see when current is flowing through the battery. If the sensed voltage is above this threshold, the duty cycle is set to 0 so that no current flows from the buck converter through the battery, and the sensed voltage is again checked. If the voltage is above 12.8, the duty cycle remains at 0 until the voltage drops below 12.6. Otherwise, the duty cycle returns to the previous operating point, and operation continues at normal until one minute has passed, then the charge-control feature is run again.

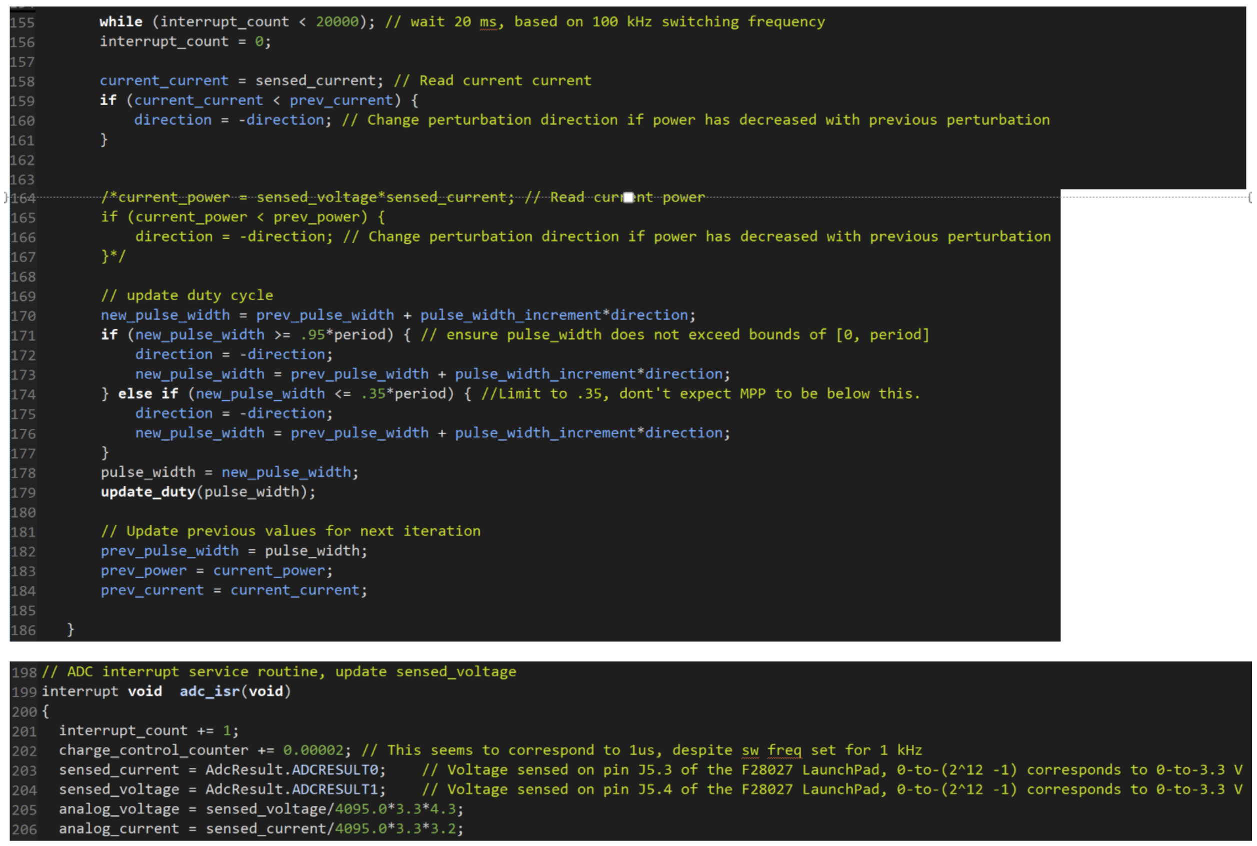
The MPPT code including the charge-control feature is shown below.



```
float interrupt_count;                                  // Tracks interrupts, acts as a clock where 1 count occurs once every switching peri
float analog_voltage;                                   // Estimated analog voltage value
float analog_current;                                   // Estimated analog current value
float charge_control_counter;                           // Acts as a clock for when to check charge control again
```



```
// ************** End of Setup *****************************************

// ************** Perturb and Iterate duty cycle **********************

   // Initialize parameters
   pulse_width = 300;   // PWM duty cycle is D = pulse_width/period. Set to initial value slightly less than expected
                        // Mpp-correspondant value.
   update_duty(pulse_width);


   int direction = 1;   // Positive indicates a positive perturbation, negative a negative perturbation
   int prev_power = 0;  // This will cause an initial increment in positive direction, to start off the algorithm
   int prev_pulse_width = pulse_width;
   int pulse_width_increment = 15;
   int current_power = 0;
   int current_current = 0;
   int prev_current = 0;
   int new_pulse_width = 0;
   charge_control_counter=60;
   int pulse_width_hold=0;
   float Vsense_scale_factor = 3603.0/3517.0; // Voltage sense output off by this factor


   interrupt_count = 0;

   while (1) {

       //Charge control
       if (charge_control_counter >= 60) { // Only check voltage once every minute.
           charge_control_counter=0;
           if (sensed_voltage >= 13.0*Vsense_scale_factor/4.3*4095/3.3) { // Allow a higher threshold than 12.8 to account for battery ESR.
                      pulse_width_hold = pulse_width;
                      pulse_width=0; // Set pulse_width to zero, and remeasure battery voltage so no ESR factor
                      update_duty(pulse_width);

                      interrupt_count=0;
                      while (interrupt_count < 20000); // Wait to settle
                      interrupt_count=0;

                      // If Battery voltage still higher, keep at zero. Otherwise, continue operation.
                      if (sensed_voltage >= 12.8*Vsense_scale_factor/4.3*4095/3.3) {
                          // Wait for battery voltage to drop below 12.6 with no current.
                          while (sensed_voltage >= 12.6*Vsense_scale_factor/4.3*4095/3.3);
                          pulse_width = pulse_width_hold;
                          update_duty(pulse_width); // Reset to previous pulse width
                      } else {
                          pulse_width = pulse_width_hold;
                          update_duty(pulse_width); // Reset to previous pulse width
                      }
                  }
       }
```

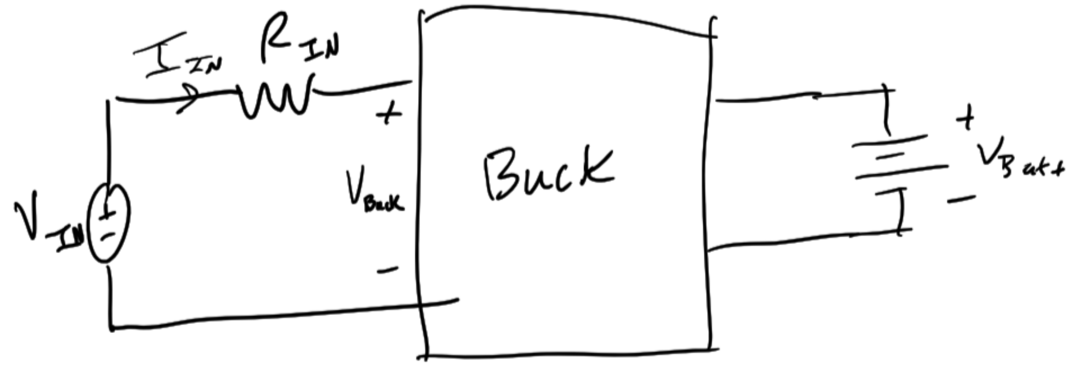


```c
    while (interrupt_count < 20000); // wait 20 ms, based on 100 kHz switching frequency
    interrupt_count = 0;

    current_current = sensed_current; // Read current current
    if (current_current < prev_current) {
        direction = -direction; // Change perturbation direction if power has decreased with previous perturbation
    }


    /*current_power = sensed_voltage*sensed_current; // Read current power
    if (current_power < prev_power) {
        direction = -direction; // Change perturbation direction if power has decreased with previous perturbation
    }*/

    // update duty cycle
    new_pulse_width = prev_pulse_width + pulse_width_increment*direction;
    if (new_pulse_width >= .95*period) { // ensure pulse_width does not exceed bounds of [0, period]
        direction = -direction;
        new_pulse_width = prev_pulse_width + pulse_width_increment*direction;
    } else if (new_pulse_width <= .35*period) { //Limit to .35, dont't expect MPP to be below this.
        direction = -direction;
        new_pulse_width = prev_pulse_width + pulse_width_increment*direction;
    }
    pulse_width = new_pulse_width;
    update_duty(pulse_width);

    // Update previous values for next iteration
    prev_pulse_width = pulse_width;
    prev_power = current_power;
    prev_current = current_current;

}
```

```c
// ADC interrupt service routine, update sensed_voltage
interrupt void  adc_isr(void)
{
  interrupt_count += 1;
  charge_control_counter += 0.00002; // This seems to correspond to 1us, despite sw freq set for 1 kHz
  sensed_current = AdcResult.ADCRESULT0;    // Voltage sensed on pin J5.3 of the F28027 LaunchPad, 0-to-(2^12 -1) corresponds to 0-to-3.3 V
  sensed_voltage = AdcResult.ADCRESULT1;    // Voltage sensed on pin J5.4 of the F28027 LaunchPad, 0-to-(2^12 -1) corresponds to 0-to-3.3 V
  analog_voltage = sensed_voltage/4095.0*3.3*4.3;
  analog_current = sensed_current/4095.0*3.3*3.2;
```

Figure 5: MPPT and Charge Control Code

This setup was tested using a power supply in series with a power resistor to emulate the voltage and current conditions of the PVpanel operating at Mpp. A $V_{IN} = 34.4$ V supply was connected in series with a rheostat set to $R_{IN} = 3.47\ \Omega$. These operating conditions were determined using the following calculations:



$P = V_{Buck} \cdot I_{IN}$, $V_{Buck} = V_{IN} - I_{IN} R_{IN}$

$\hookrightarrow P = V_{IN} I_{IN} - I_{IN}^2 R_{IN}$

at max power point, $\frac{dP}{dI} = V_{IN} - 2 I_{IN} R_{IN} = 0$

$\hookrightarrow \frac{V_{IN}}{2 I_{IN}} = R_{IN} \rightarrow I_{IN} \cdot R_{IN} = \frac{V_{IN}}{2}$

To get $V_{Buck} = 17.2\ V\ (V_{mpp})$, $V_{IN} = 34.4\ V$

To get $I_{IN} = 4.95\ A$, $R_{IN} = 3.47\ \Omega$

Figure 6: Determination of V_{IN} and R_{IN}

This input setup was connected, along with the discharged battery at the output. The goal was to see the MPPT code regulate the duty cycle such that the input current was around 4.95 A, which would cause a roughly 17.2 V drop across the input rheostat, and thus result in 17.2 V to be provided as input voltage to the buck. Our setup successfully resulted in these conditions after some debugging of the code. The pulse_width_increment size and settle time both had to be adjusted to result in successful MPP tracking. Additionally, at first the code was maximizing output power, instead of output current, and this resulted in a low duty cycle, around 45%. The battery dynamics caused this implementation to not work well to operate the supply at the maximum power point, since low duty cycle causes low input power on the PV I-V curve. Changing the code to maximize battery current resulted in much better operation, with duty cycle ranging from 85 - 90%.

The resulting MPP operation values, along with corresponding oscilloscope capture, are shown below:

Duty Cycle	Input Voltage	Input Current	Input Power	Output Voltage	Output Current	Output Power	Efficienc y
87%	17.50 V	4.80 A	83.98 W	13.74 V	5.74 A	73.13 W	87.08%

Table 1: MPP tracked operation with power supply input

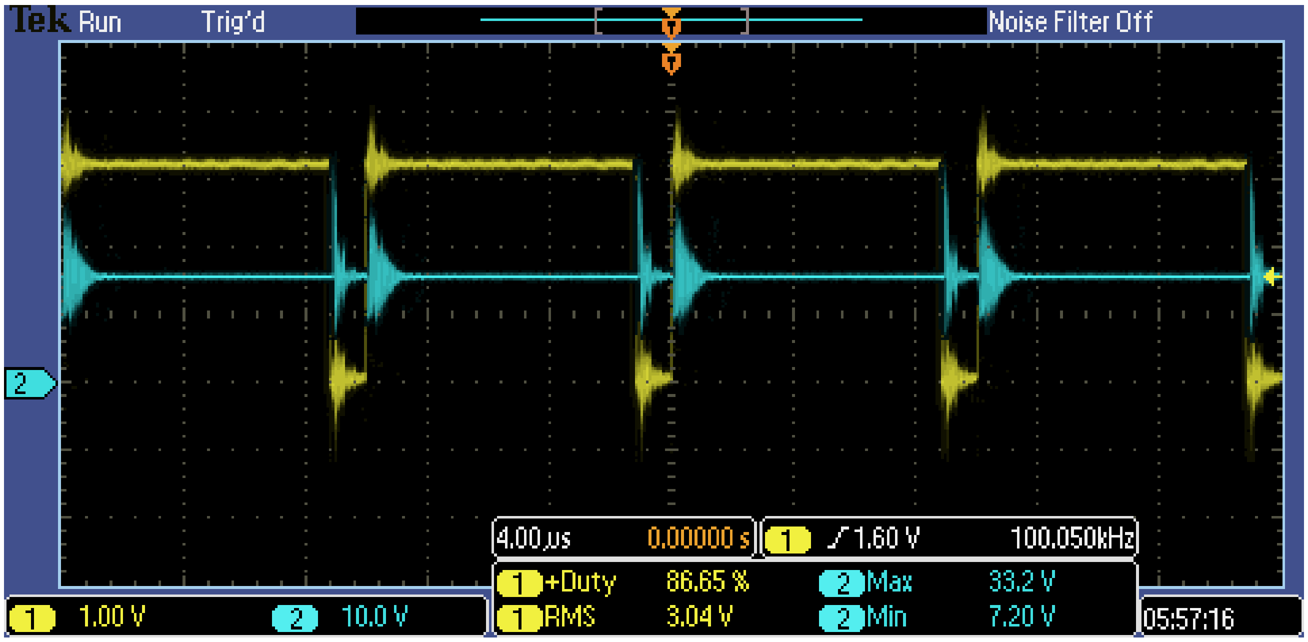


Figure 7: Oscilloscope waveform of converter input voltage (blue) and V_{PWM} signal while tracking peak power point

In order to verify that the bench test system was operating at the maximum power point, we operated the system with MPPT turned off, and instead varied duty cycle manually. The resulting curve (figure 8) shows that the maximum power duty cycle is in the region of 87% duty cycle.

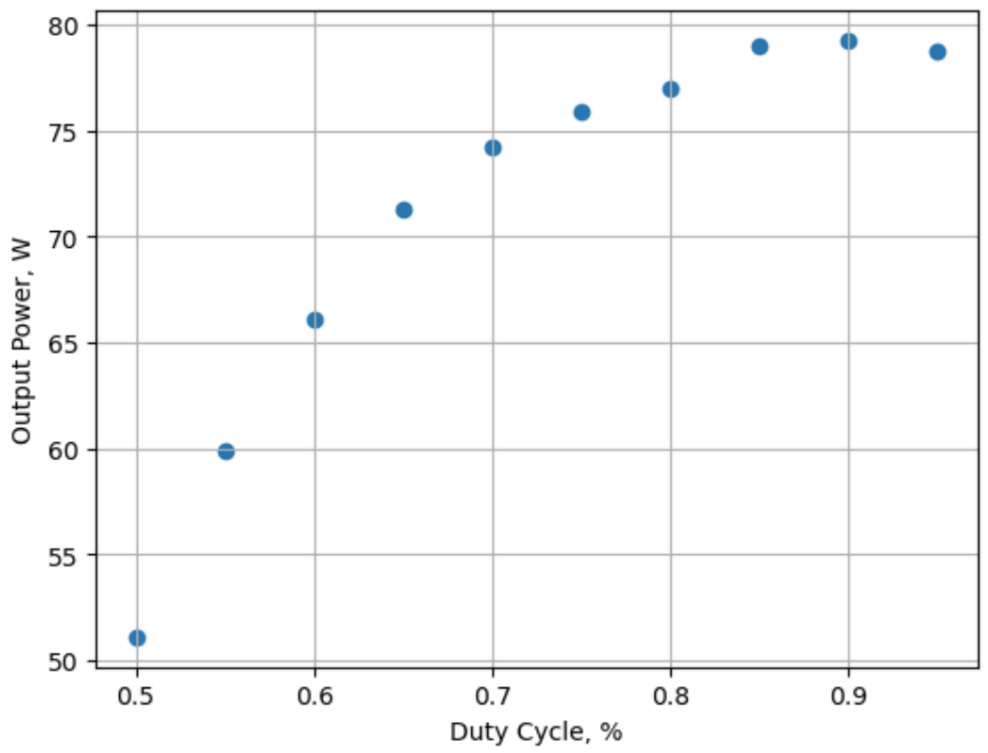


Figure 8: Power curve vs manually varied duty cycle on bench test setup

Finally, the charge control feature was validated to work by two methods. The first was running the converter with a power supply directly connected to the input, and a 3.5 Ω power resistor at the output. The MPPT algorithm maximized duty cycle with a purely resistive output, and the input voltage was set to just above 13.1 V, the threshold voltage of the initial entry into the charge control feature. Since dropping the duty cycle to zero caused 0 V output, the charge control feature detected a voltage less than 12.8 V after decreasing duty cycle, and allowed continued operation of MPPT. The second method of validation was connecting a power supply directly to the output, with no input. The output voltage was increased above 13.1 V, which caused the charge control loop to set duty cycle to zero to check if voltage would drop below 12.8 V. Since duty cycle did not control the output voltage in this scenario, the sensed voltage remained above 12.8 V, and the charge control loop kept the duty cycle at zero until the power supply voltage was reduced below 12.6 V. MPPT operation continued as normal after this.

Step 4: Complete System Test

The MPPT buck converter system was tested with the PV panel as an input on a sunny day. The system was tested at 3:15 PM on 3/06/2023. Solar irradiance was 880 W/m^2. Waveforms for the unshaded PV panel output voltage and v_{pwm} at the MPP are shown below.

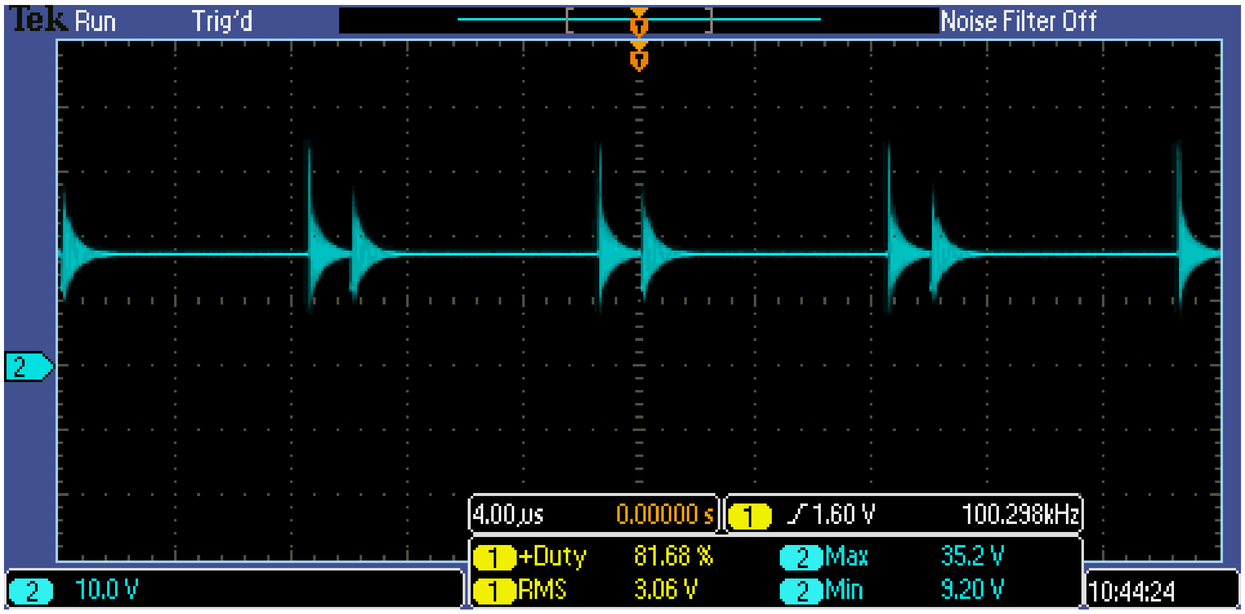


Figure 9: Oscilloscope waveform of the PV panel output voltage

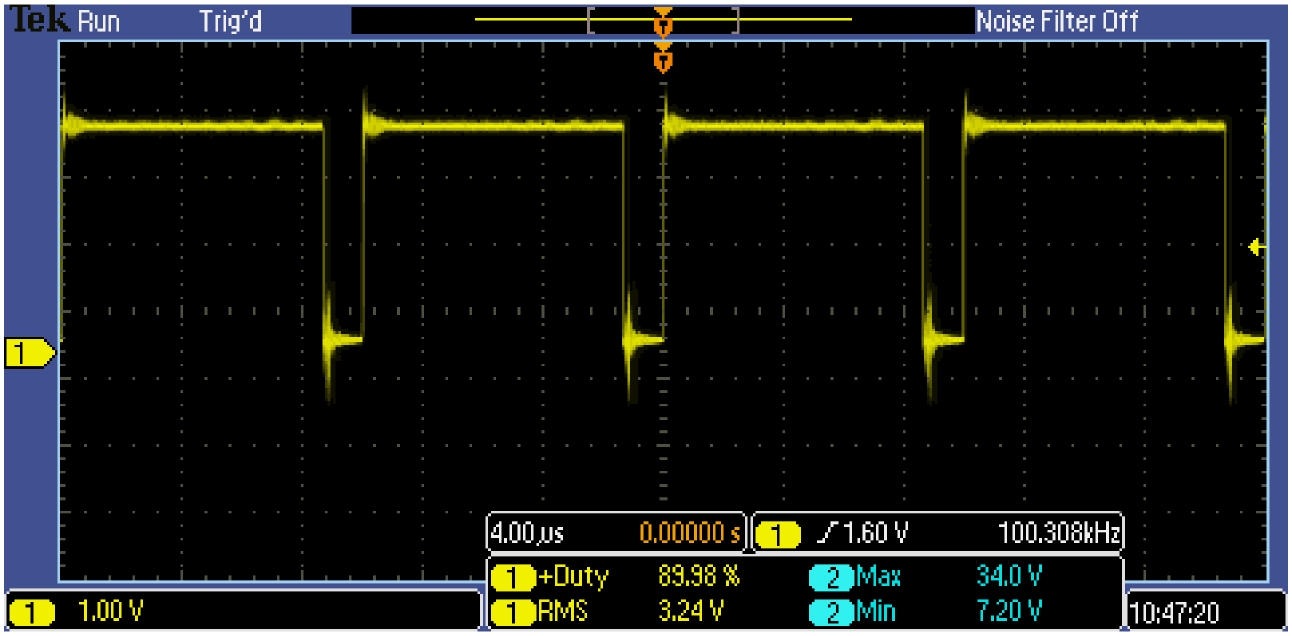


Figure 10: Oscilloscope waveform of the v_{pwm} output of the Launchpad

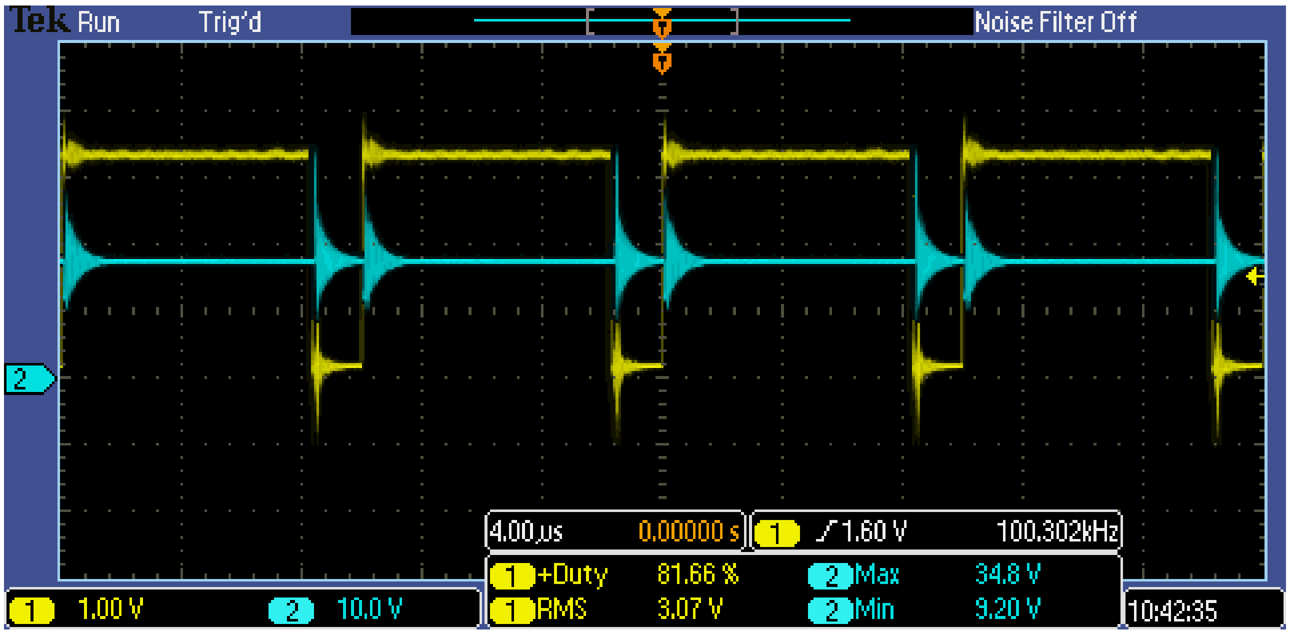


Figure 11: Oscilloscope waveforms of the PV panel output and v_{pwm} overlaid, with the PV panel output voltage shown in blue and v_{pwm} shown in yellow.

The input and output voltages and currents of the Buck converter were recorded using multimeters. They are shown below, along with the calculated input and output power.

V_{in} = 15.287 V
I_{in} = 5.721 A
P_{in} = 87.56 W
V_{out} = 13.115 V
I_{out} = 5.22 A
P_{out} = 68.46 W
Efficiency = P_{in} / P_{out}
Efficiency = 0.78 = 78%
Duty Cycle = 85-92.5%

The Duty Cycle of the Buck Converter was close to the MPP Duty Cycle we found in the lab under simulated STC, however the efficiency of the Buck Converter was fairly low. At 87.56 W and a Voltage of 13.115 V, if our converter was operating at 100% efficiency the current should be 6.76 A. Our Converter operated close to the MPP, but was not perfect. The overall efficiency of the PV panel and Buck Converter with MPPT can be found by converting the solar irradiance data to Watts, and then comparing to the Output power of the Buck Converter.

$$68.46 \text{ W} / (880 \text{ W/m}^2 * (1.2 \text{ m} * 0.527 \text{ m})) = 12.3\%$$

We did not take DET measurements on the same day as our MPPT measurements were taken, so direct comparison of the power supplied to the battery would not be useful(unless we were able to take them on a day with the same irradiance). However, in Experiment 2 we measured the Direct Energy Transfer from the PV panel to the battery to be 32.23 W, under irradiance conditions of 550 W/m^2. This yielded an efficiency of

$$32.23\text{ W} / (550\text{ W/m}^2 * (1.2\text{m} * 0.527\text{ m})) = 9.23\%$$

The theoretical efficiency in the panel specifications at irradiance of 800 W/m^2 is 12.5%. This shows that our MPPT algorithm and Buck Converter charge the Battery near the maximum theoretical efficiency of the PV panel, and at a much higher efficiency(over 3%) than through Direct Energy Transfer. The observed Duty Cycle also corresponds to the Maximum Power Point Duty Cycle we measured when manually varying the Duty Cycle in Step 3.

MPPT with Four Cells Shaded

To test our MPPT under shaded conditions, we shaded four cells and then recorded the input and output voltages and currents of the Buck Converter. The voltages and currents, along with the calculated power, are shown below:

V_{in} = 13.43 V
I_{in} = 0.201 A
P_{in} = 2.70 W
V_{out} = 12.3 V
I_{out} = 0.215 A
P_{out} = 2.64 W
Efficiency = P_{in} / P_{out}
Efficiency = 0.98 = 98%
Duty Cycle = 70-90%

Direct Energy Transfer Data with four cells shaded was recorded on 3/9/2023 at 3:00 P.M. with a Solar Irradiance of 966 W/m^2, shown below:

V_{in} = 13.4 V
I_{in} = 0.29 A
P_{in} = 3.89 W
V_{out} = 12.2 V
I_{out} = 0.26 A
P_{out} = 3.17 W
Efficiency = 81.4%

The overall efficiency of the PV panel and Buck Converter with MPPT under shaded conditions is shown below:

$$2.64 \text{ W} / (880 \text{ W/m}^2 * (1.2 \text{ m} * 0.527 \text{ m})) = 0.47\%$$

Our Direct Energy Transfer efficiency under the irradiance of 966 W/m^2 is calculated below:

$$3.17 \text{ W} / (966 \text{ W/m}^2 * (1.2\text{m} * 0.527 \text{ m})) = 0.52\%$$

The Buck Converter Duty Cycle varied between 0.7 and 0.9 during this testing. This is a wide range, but still roughly centered around our tested max duty cycle under unshaded conditions. Our Buck Converter operated at 98% efficiency under shaded conditions, likely due to the low current reducing losses in the Buck Converter components, and more efficient than the 81.4% efficiency of Direct Energy Transfer. However, in terms of efficiency of the energy harvested from the available solar energy, under shaded conditions our Direct Energy Transfer was more efficient than our Buck Converter with MPPT.

Possible Improvements:
Under partially shaded conditions, multiple peaks can be present in the P-V curve of the panel. Our perturb-and-observe MPPT algorithm only finds the local peak, which under unshaded conditions is sufficient as there is only one peak in the P-V curve. Changing our MPPT code to periodically sweep the duty cycle can help us to find which local peak is the true MPP peak.