University of Colorado
Boulder

Photovoltaic Power Electronics
ECEA 5718 Battery Management Laboratory

Battery Management

Kai I. Tam

April 08, 2022

1. Charge taper mode in MATLAB/Simulink model

- Report documents the working controller code that implements MPPT and charge taper modes in the MATLAB/Simulink model (15 points)

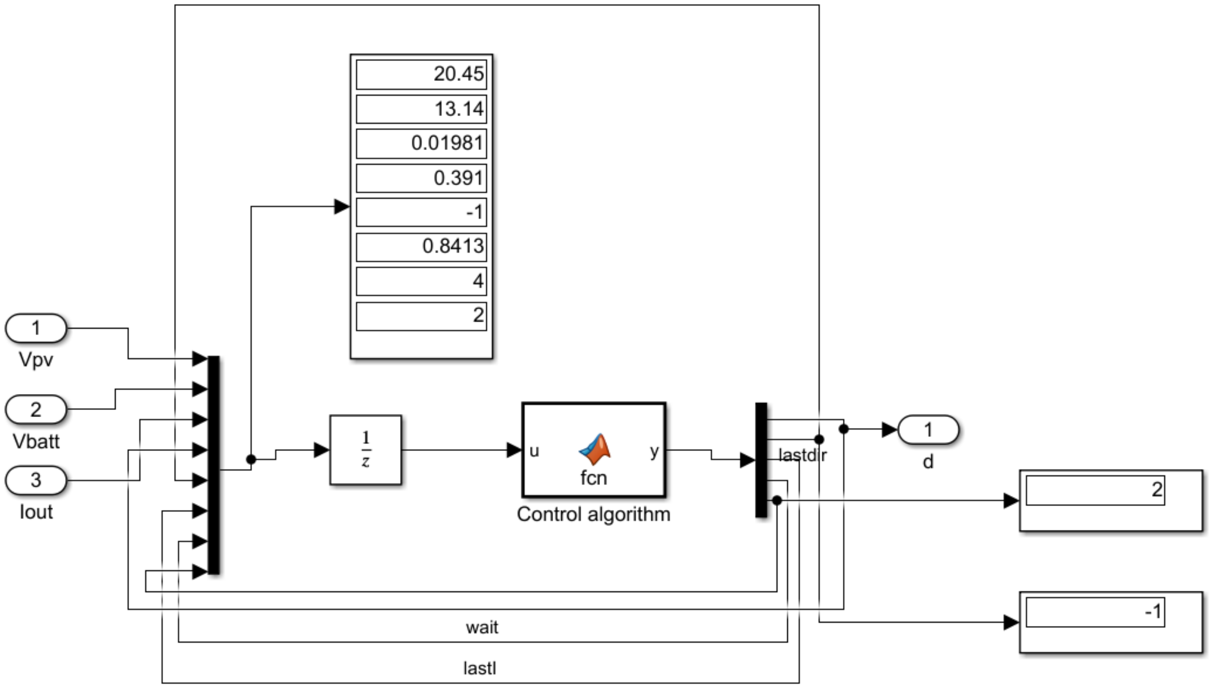


Figure 1. Controller demo readings after 60s simulation.

```
%% input signals (sampled)
%
    Vpv = u(1);             % sensed PV voltage (might or might not be used in MPPT algorithm)
    Vbatt = u(2);           % sensed battery voltage (needed for charge taper mode)
    Iout = u(3);            % sensed SEPIC output current (might be used in MPPT algorithm)
    lastduty = u(4);        % MPPT last duty cycle
    lastdir = u(5);         % MPPT last direction: +1 or -1 (+1 means duty cycle was increased)
    lastI = u(6);           % MPPT last converter output current value
    wait = u(7);            % variable used to count settling periods after change in duty cycle
    lastmode = u(8);
%
%% Algorithm parameters, can be adjusted to tune control performance
%
% Define your algorithm parameters here
    settlePeriods = 10;     % number of sample periods to wait for system to settle
    step = 0.001;           % MPPT algorithm duty cycle step size
    Imin = 0.1;             % threshold to terminate charge taper mode
    Dmax = 0.9;             % limit max duty cycle of SEPIC
    Vtaper = 13.1;          % Charge taper: regulate Vbatt = Vtaper in charge taper mode
    Vh = 0.06;              % voltage hysteresis to prevent cycling between modes
    Vfloat = 13.5;
%
```

Matlab 1. Input and Define.

```matlab
%% Control algorithm
%
% Enter your control algorithm here
    if wait <=0
        wait = settlePeriods;                                % when down counter is up, execute below

        if Vbatt < Vtaper                                    % MPPT, (when Vbatt lower than Vtaper)
            mode = 1;                                        % setting mode as 1 (MPPT)
            if Iout < lastI                                  % comparing Last Iout to current Iout
                direction = -lastdir;                        % if last Iout larger than Iout, change direction.
            else
                direction = lastdir;                         % if last Iout smaller than Iout, change direction.
            end
            d = min(Dmax, max(0,lastduty+direction*step));   % +/-d according to direction
            lastI = Iout;                                    % store Iout state to next reading state.
        else                                                 % charge taper, (when Vbatt higher than Vtaper)
            mode = 2;                                        % setting mode as 2 (Charge Taper)
            if Vbatt > Vtaper + Vh                           % compare Vbatt to Vtaper + Vh (hysteresis)
                direction = -1;                              % when Vbatt higher than Vtaper + Vh, charge battery with decrement of d
            else
                direction = 0;                               % when Vbatt lower than Vtaper + Vh, charge battery in constant d
            end
            d = min(Dmax, max(0,lastduty+direction*step));   % -d or remaing the same d according to direction
            direction = lastdir;                             %% important, resetting direction state incase jump back to MPPT
        end
    else
        wait = wait-1;                                       % decrement wait (count variable)
        d = lastduty;                                        % pass duty cycle to next sampling period
        direction = lastdir;                                 % pass direction to next sampling period
        mode = lastmode;                                     % pass mode to next sampling period (monitoring purpose only)
    end
%
% vector output of this function
y = [d direction lastI wait mode];
```

Matlab 2. MPPT / Charge Taper

Most of the code from MPPT are repurposed on this project, adding an "IF" statement comparing the current Battery Voltage (***Vbatt***) to $V_{threshold}$ (***Vtaper***).
The controller algorithm changed from MPPT (Mode 1) to Charge Taper (Mode 2) when ***Vbatt*** > ***Vtaper***. At this instant, the controller keeps a constant duty cycle charging (d = 0.432). Notice that the Iout envelope disappeared at 6.4s (Figure 2)
Duty Cycle starting to decrement when ***Vbatt*** > ***Vtaper + Vh*** at 7.5s (Figure 2). Notice that Battery Voltage is keeping constant, charging current is decreasing gradually and SOC increasing gradually.

- Report documents the simulation waveforms of battery voltage, battery current, and duty cycle for operation in the charge taper mode (15 points)
- Report documents the transition from MPPT mode to charge taper mode in the MATLAB/Simulink simulation. (5 points)

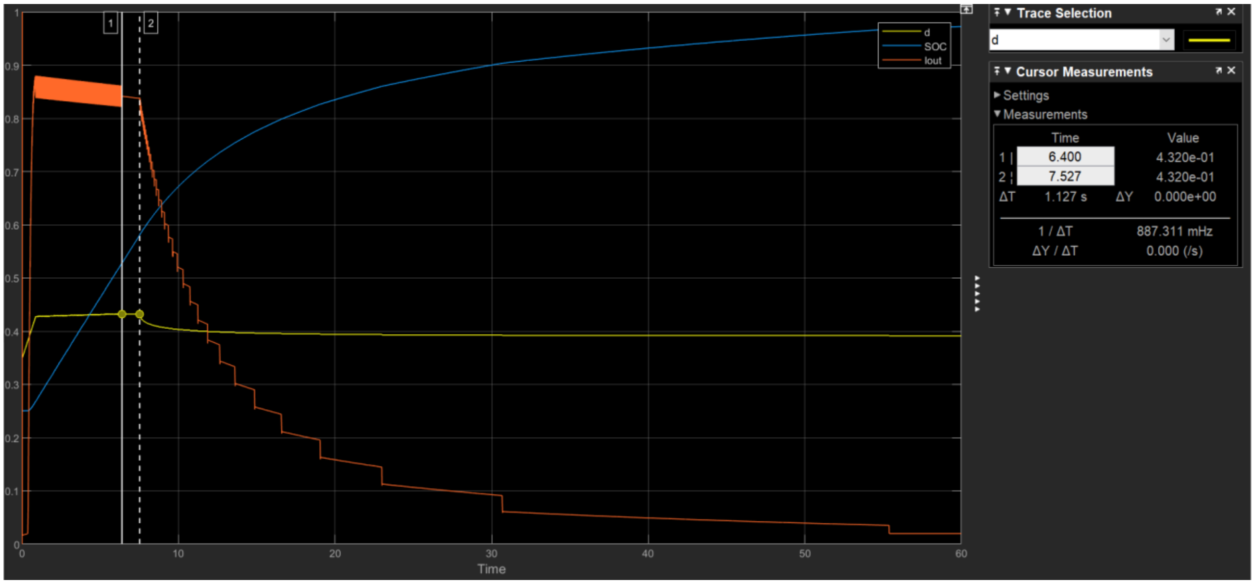


Figure 2. d SOC Iout (1 minute simulation time)

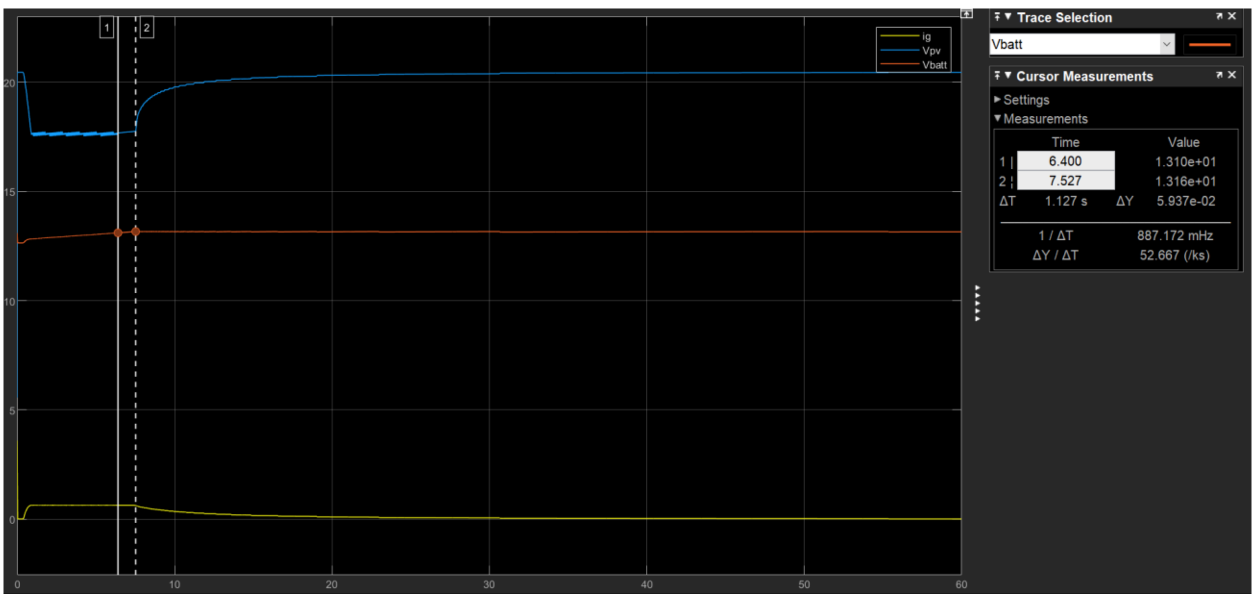


Figure 3. Ig Vpv Vbatt (1 minute simulation time)

MPPT Mode:	0 - 6.4s
Charge Taper Mode:	6.4s - 60s

- Report documents the transition from charge taper mode to MPPT mode when the battery becomes sufficiently discharged. (5 points)

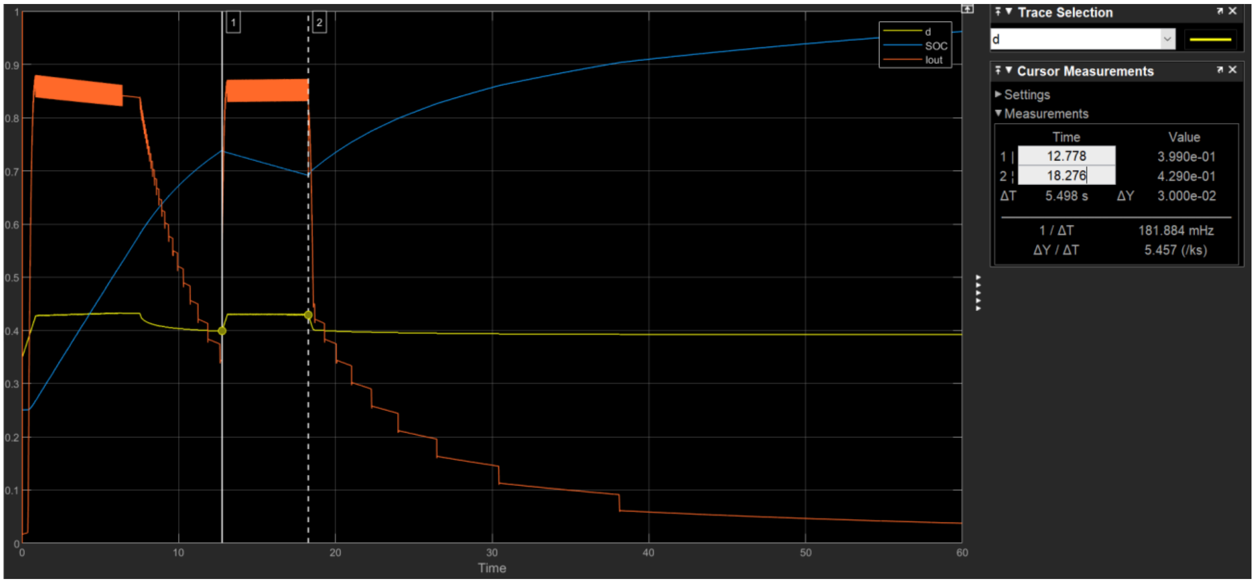


Figure 4. d SOC Iout (1 minute simulation time) (1A step load)

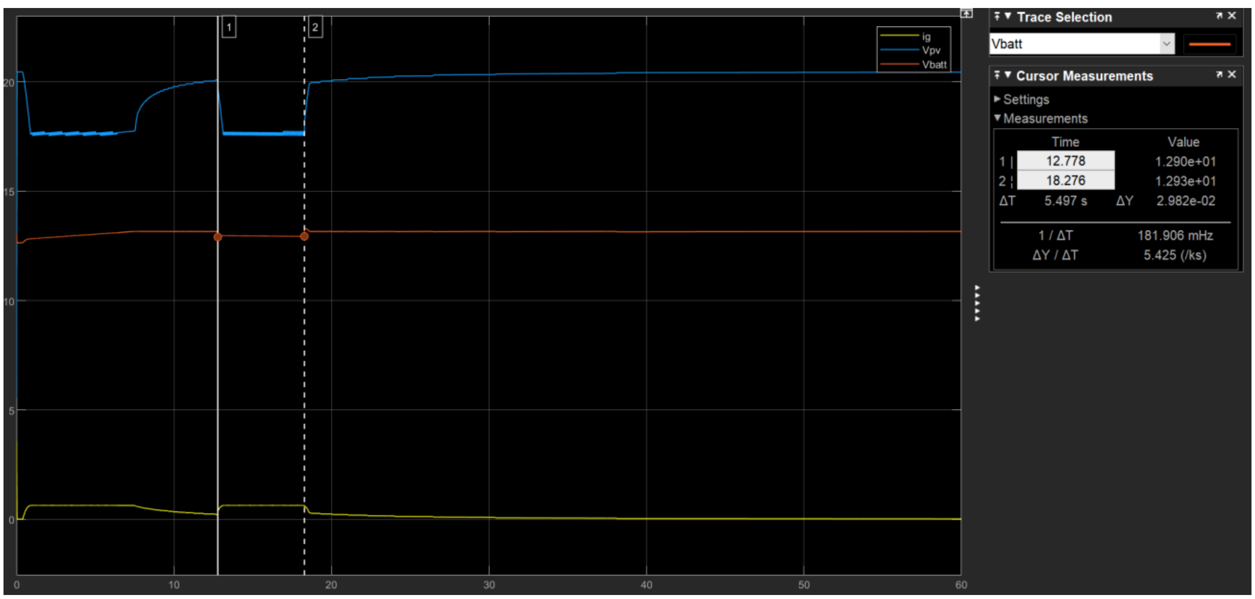


Figure 5. Ig Vpv Vbatt (1 minute simulation time) (1A step load)

A 1A load current is manually applied between simulation time 12.778s to 18.276s. Battery Voltage (***Vbatt***) decreases down to 12.90V, notice that Duty Cycle jumps back from 0.399 to 0.429, charging current increased back to average of 0.85A, the controller is working in MPPT (Mode 1).
When the load is lifted, the Battery Voltage (***Vbatt***) jumps back up to ~13.15V, resuming back to Charge Taper mode (Mode 2), Duty Cycle decreases and maintaining our 13.16V $V_{threshold}$.

2. Float mode in MATLAB/Simulink model

- Report documents the working controller code that implements three-mode (MPPT/charge taper/float) control in the MATLAB/Simulink model (5 points)

```
function y = fcn(u)
% Battery ESR vs SOC
SOC = u(1);
ibatt = u(2);
%ESR0 = 0.15;   % minimum ESR
ESR0 = 0.032;   % changed to 0.032 (Matching Datasheet)
%S0 = 1.1;      % value of SOC that causes ESR to go to infinity (empirical parameter)
S0 = 1.0;       % changed to 1.0 to prevent charge over 1.0 SOC
if ibatt > 0
    ESR = ESR0/(1-SOC/S0); % charging
else
    ESR = ESR0/(1-(1-SOC)/S0); % discharging
end
y = ESR;
```

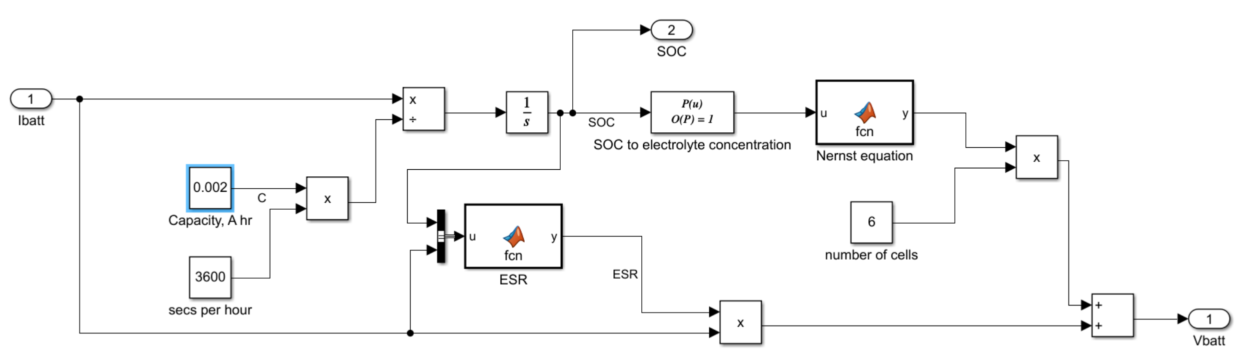
Matlab 3. Battery ESR Model.

I modified the Battery ESR model to get a more realistic simulation, ESR0 changed to 32mΩ [1], S0 changed to 1.0 to prevent charge over 1.0 SOC

```
%% Algorithm parameters, can be adjusted to tune control performance
%
% Define your algorithm parameters here
    settlePeriods = 10;      % number of sample periods to wait for system to settle
    step = 0.001;            % MPPT algorithm duty cycle step size
    Imin = 0.05;             % threshold to terminate charge taper mode
    Dmax = 0.9;              % limit max duty cycle of SEPIC
    Vtaper = 13.1;           % Charge taper: regulate Vbatt = Vtaper in charge taper mode
    Vh = 0.04;               % Vbatt >= Vtaper + Vh, controller goes into Float mode
    Vfloat = 13.16;          % Vbatt >= Vfoat, controller further cuts off duty cycle
%
```

Matlab 4. Define Parameters for final design

By adding another operating mode, I also modified the define parameters to get a more realistic SOC simulation, Vh changed from 0.06 to 0.04.



Capacity of Battery is changed to 0.002, to get three Charging modes simulated in 60s time.

```
29  %% Control algorithm
30  %
31  % Enter your control algorithm here
32      if wait <=0
33          wait = settlePeriods;                              % when down counter is up, execute below
34  
35          if Vbatt < Vtaper                                  % MPPT, (when Vbatt lower than Vtaper)
36              mode = 1;                                      % setting mode as 1 (MPPT)
37              if Iout < lastI                                % comparing Last Iout to current Iout
38                  direction = -lastdir;                      % if last Iout larger than Iout, change direction.
39              else
40                  direction = lastdir;                       % if last Iout smaller than Iout, change direction.
41              end
42              d = min(Dmax, max(0,lastduty+direction*step));  % +/-d according to direction
43              lastI = Iout;                                  % store Iout state to next reading state.
44          else                                               % Charge Taper/Float , (when Vbatt higher than Vtaper)
45              if Iout > Imin                                     % when Charging current Iout higher than Imin, execute mode 2 below
46                  mode = 2;                                      % setting mode as 2 (Charge Taper)
47                  if Vbatt > Vtaper + Vh                         % compare Vbatt to Vtaper + Vh (hysteresis)
48                      direction = -1;                            % when Vbatt higher than Vtaper + Vh, charge battery with decrement of d
49                  else
50                      direction = 0;                             % when Vbatt lower than Vtaper + Vh, charge battery in constant d
51                  end
52                  d = min(Dmax, max(0,lastduty+direction*step));  % -d or remaing the same d according to direction
53                  direction = lastdir;                           %% important, resetting direction state incase jump back to MPPT
54              else                                               % when Charging current Iout less than Imin, execute mode 3 below
55                  mode = 3;                                      % setting mode as 3 (Float)
56                  if Vbatt >= Vfloat                             % compare Vbatt to Vfloat
57                      direction = -1;                            % when Vbatt higher/equalt to Vfloat, charge battery with decrement of d
58                  else
59                      direction = 0;                             % when Vbatt lower/equalt to Vfloat, charge battery in constant d
60                  end
61                  d = min(Dmax, max(0,lastduty+direction*step));  % -d or remaing the same d according to direction
62                  direction = lastdir;                           %% important, resetting direction state incase jump back to MPPT
63              end
64          end
65      else
66          wait = wait-1;                                     % decrement wait (count variable)
67          d = lastduty;                                      % pass duty cycle to next sampling period
68          direction = lastdir;                               % pass direction to next sampling period
69          mode = lastmode;                                   % pass mode to next sampling period (monitoring purpose only)
70      end
71  %
72  % vector output of this function
73  y = [d direction lastI wait mode];
```

Matlab 5. MPPT / Charge Taper / Float

Adding another condition to further define Float mode.

In Line 45, when charging current is larger than a threshold ***Imin***, it would execute as Charge Taper (Mode 2), keeping constant Duty Cycle Charging, once the Battery Voltage charge higher than Vtaper + Vh (13.14V), decrement Duty Cycle and maintaining the constant voltage.

In Line 54, When Charging current (***Iout***) decreases down below ***Imin***, the controller goes into Float Mode (Mode 3), it would keep Duty Cycle in constant until it charges up to Vfloat (13.16V, by choice, which not over charging above 1.0 SOC), once Battery Voltage charged above or equal to Vfloat, decrement Duty Cycle until it completely cut off.

- Report documents the transition from charge taper mode to float mode, in the simulation (5 points)

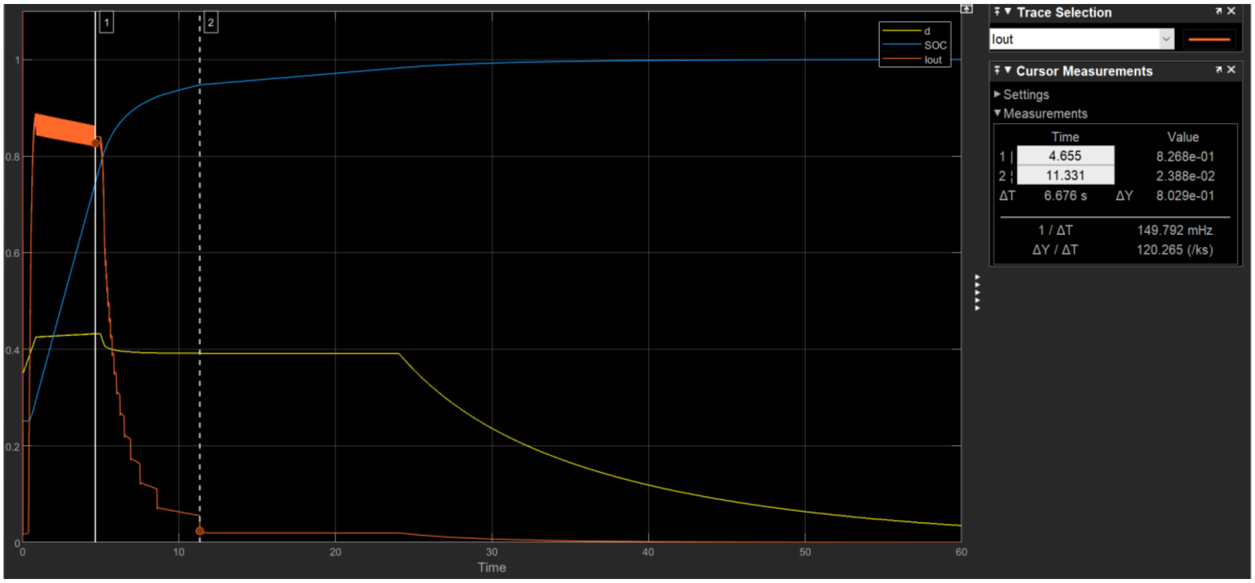


Figure 6. d SOC Iout (1 minute simulation time)

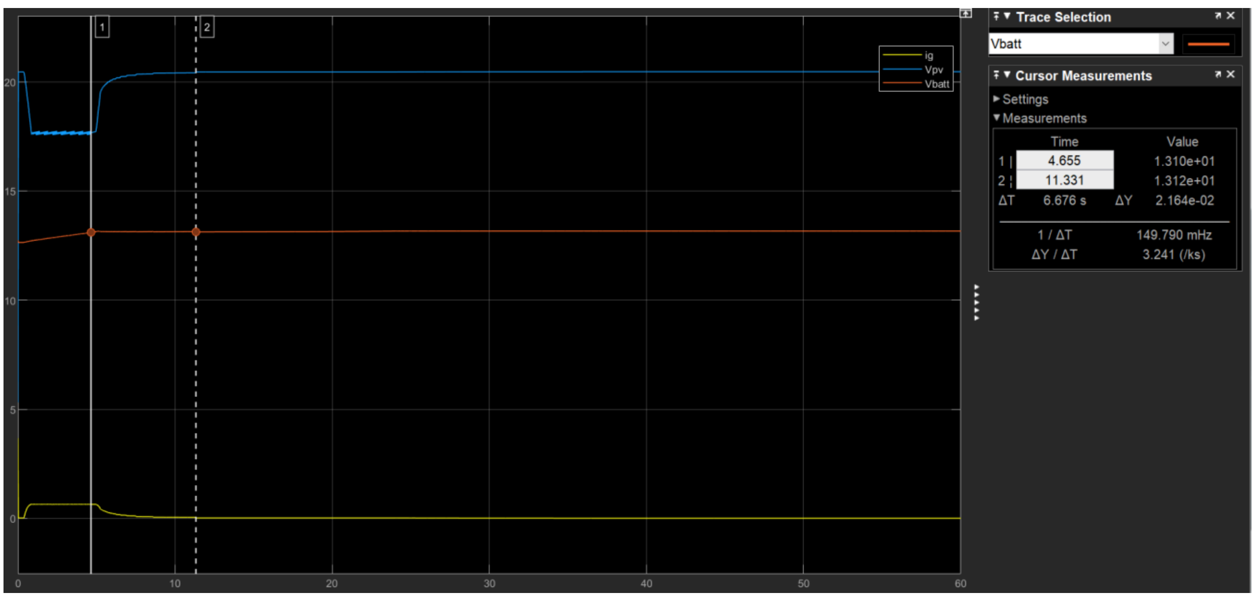


Figure 7. Ig Vpv Vbatt (1 minute simulation time)

MPPT Mode: 0 - 4.655s
Charge Taper Mode: 4.655s – 11.331s
Float Mode: 11.331s - inf

Controller Duty Cycle starts to roll off at around 24s, when Battery Voltage charge up to 13.16V.

- Report documents the transition from float mode to charge taper mode, in the simulation (5 points)

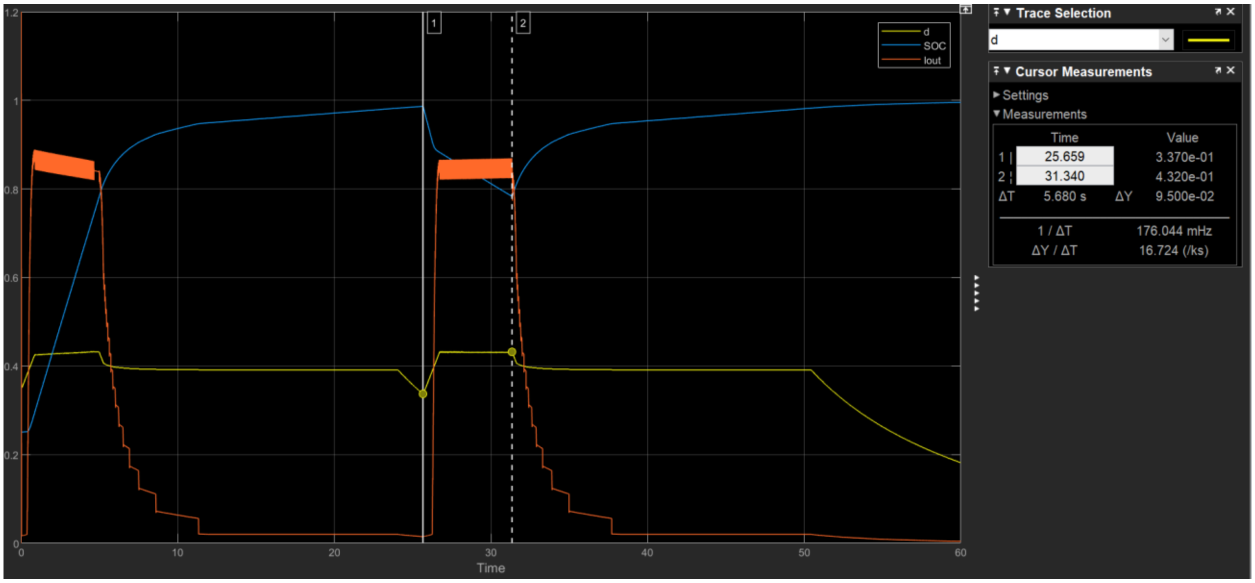


Figure 8. d SOC Iout (1 minute simulation time) (1A step load)

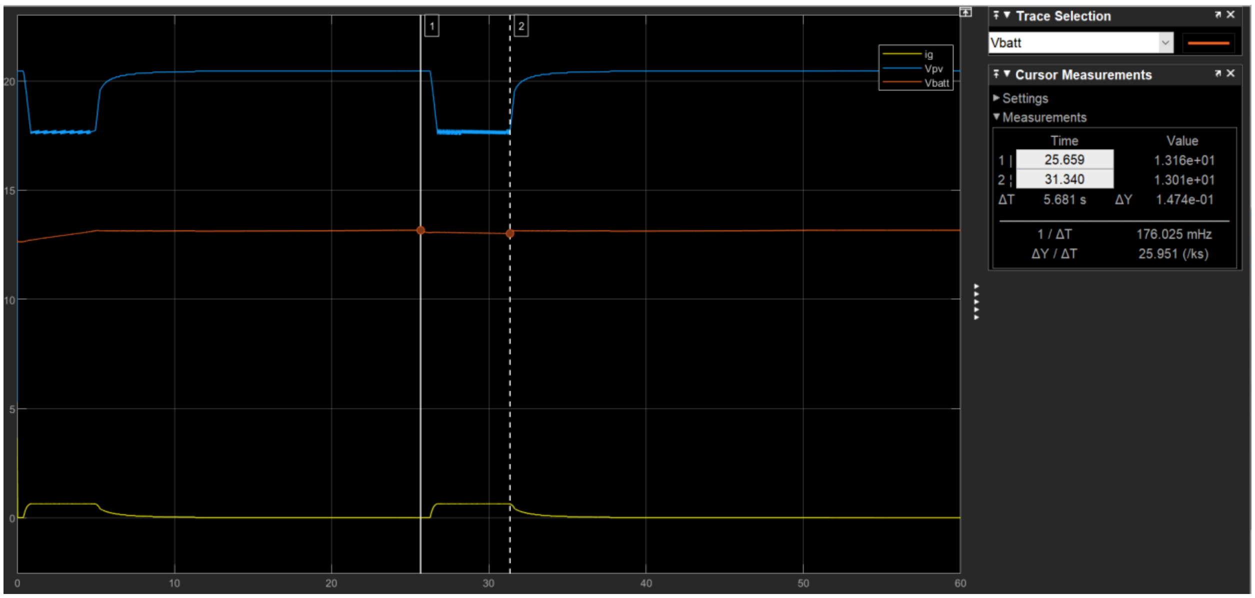


Figure 9. Ig Vpv Vbatt (1 minute simulation time) (1A step load)

A 1A load current is manually applied between simulation time 25.659s to 31.340s. Battery Voltage (***Vbatt***) decreases down to 13.01V, notice that Duty Cycle jumps back to 0.432, charging current increased back to average of 0.85A, the controller is working in MPPT (Mode 1). SOC of battery is discharged back down to 0.8.
When the load is lifted, the Battery Voltage (***Vbatt***) jumps back up to ~13.14V, resuming back to Charge Taper mode (Mode 2), Duty Cycle start to roll off and goes into Float Mode (Mode 3) once Charging current lower than Imin.

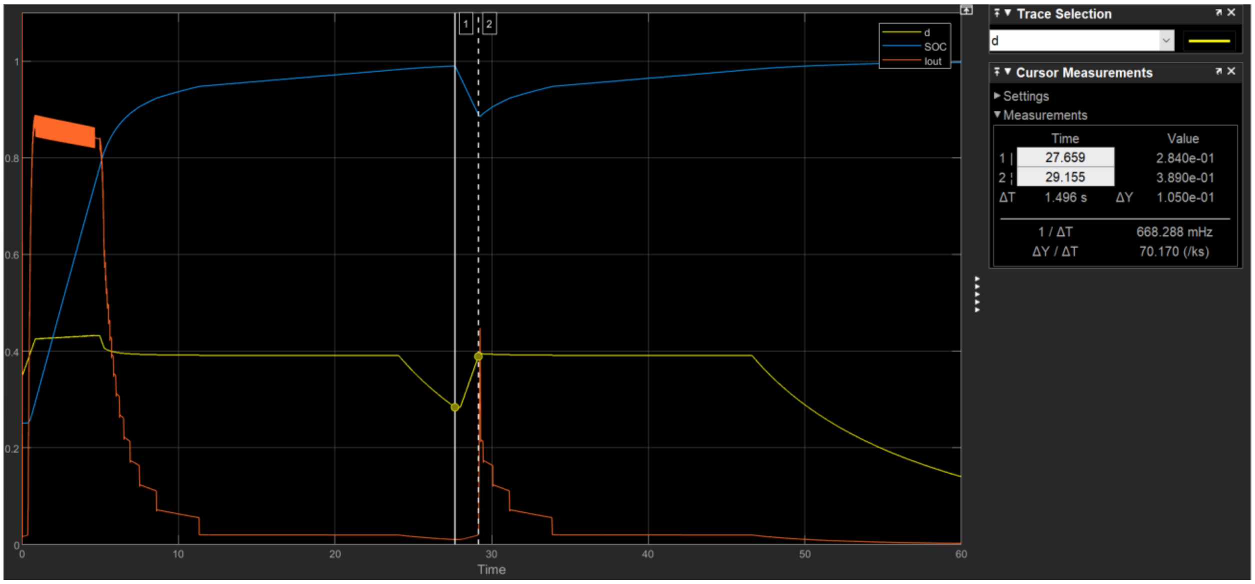


Figure 10. d SOC Iout (1 minute simulation time) (0.5A step load)

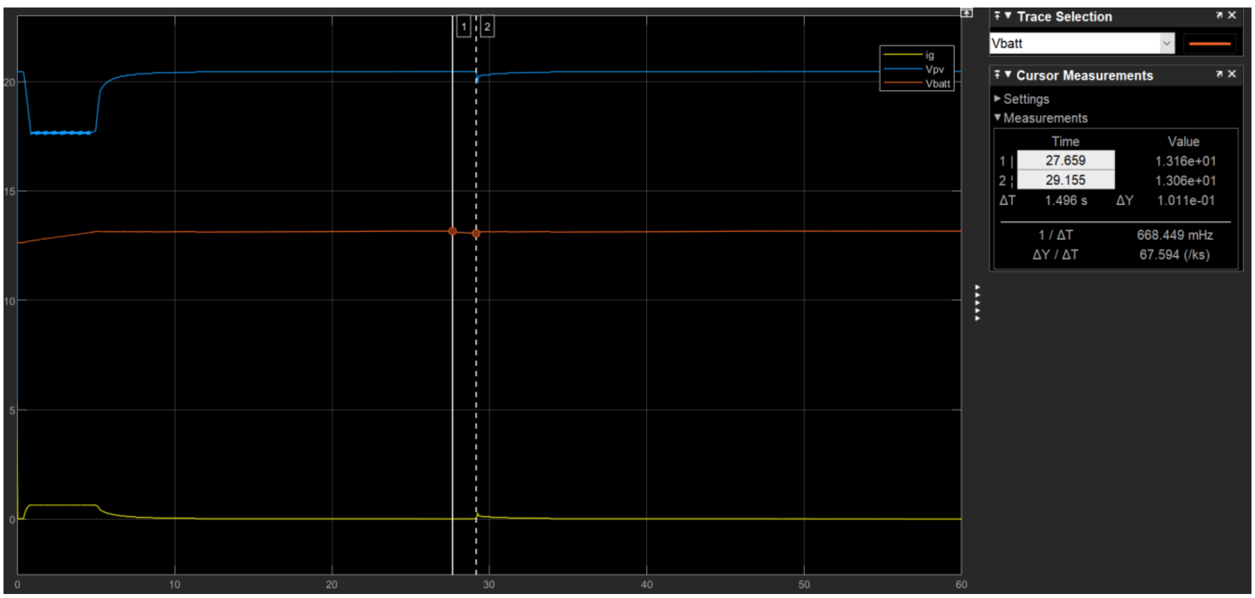


Figure 11. Ig Vpv Vbatt (1 minute simulation time) (0.5A step load)

A 0.5A load current is manually applied between simulation time 27.659s to 29.155s. Battery Voltage (***Vbatt***) decreases down to 13.12V, notice that Duty Cycle jumps back to 0.389, charging current increased slightly, the controller is working in Charge Taper Mode (Mode 2). SOC of battery is discharged back down to 0.9.
When the load is lifted, the Battery Voltage (***Vbatt***) jumps back up to ~13.14V, we are still in Charge Taper mode (Mode 2), Duty Cycle start to roll off and goes into Float Mode (Mode 3) once Charging current lower than Imin.

3. Hardware demonstration of three-mode control

- Report documents the working microcontroller code that implements three-mode (MPPT/charge taper/float) control. The relevant interrupt service routine code should be included. (15 points)

Condition				Voltage			Current		
Input V	D (%)	Real D (%)	Load (Ω)	Output (V)	Vadc	V gain	Output (A)	Aadc	A gain
17	5	5.78	30	1.549	250	161	0.052	143	2770
	10	10.74		2.360	350	148	0.079	222	2822
	15	15.71		3.277	475	145	0.109	372	3406
	20	20.68		4.375	617	141	0.146	558	3826
	25	25.65		5.679	789	139	0.189	793	4189
	30	30.62		7.201	982	136	0.240	1064	4433
	35	35.59		8.965	1223	136	0.299	1389	4648
	40	40.56		11.017	1498	136	0.367	1754	4776
	45	45.53		13.410	1805	135	0.447	2184	4886
	50	50.50		16.204	2165	134	0.540	2678	4958

Table 1. Voltage-ADC gain, Current-ADC gain

We will utilize the Voltage and current gain ratio based on the data obtained from last project.

Since the voltage of interest (average Battery Voltage) will be around 11-13V, we will use 135 as our voltage gain ratio.

We will need to define our current charging threshold ***imin*** when we are going into Float mode, Since our Current Sensor Circuit is not exactly linear in the lower end range, we will use the experiment data. I chose 0.1A as my threshold Imin rather than 0.05A in Matlab. The ADC value of 0.1A is around 372.

I also increase the value of (***vh***) to 0.37V and (***vfloat***) to 13.4V, for experimental purpose.

```
//
// Globals
//
    uint16_t                LoopCount;
    uint16_t                ConversionCount;

    uint32_t    volatile    counter             =   0;          // For extending ADC sampling time.
    uint16_t    volatile    direction           =   1;          // 1 increasing (default), 0 decreasing

    uint16_t    volatile    last_control        =   10;         // Start up duty cycle with 10.
    uint16_t    volatile    control             =   10;         // Start up duty cycle with 10.

    uint32_t    volatile    voltage;                            // Battery Voltage
    uint32_t    volatile    last_current        =   0;          // last current value for comparison
    uint32_t    volatile    current;                            // Charge / output Current ADC value

    uint16_t    volatile    increment           =   1;          // increment of Duty Cycle (1/120)

    uint16_t    volatile    vtaper              =   1755;       // 13.0V    * gain ratio (135)
    uint16_t    volatile    vh                  =   50;         // 0.37V    * gain ratio (135)
    uint16_t    volatile    vfloat              =   1809;       // 13.4V    * gain ratio (135)
    uint16_t    volatile    imin                =   372;        // around 0.1A

    uint16_t    volatile    mode                =   1;          // charging mode (monitoring purpose only)
```

Figure 12. Defining Global Variables

```
//
//  adc_isr -
//
#pragma CODE_SECTION(adc_isr, "ramfuncs");
__interrupt void
adc_isr(void)
{
        counter = counter + 1;                                       // counting up

        if (counter >= 14000)                                        // change settling time here (14000 = 40mS til next ADC reading)
            {
            // GPIO19 Toggle
                    GPIO_setHigh(myGpio, GPIO_Number_19);            // Toggle GPIO high to find out the Sampling Time.

            // Reading
                    current = ADC_readResult(myAdc, ADC_ResultNumber_1);  // Reading Battery Charging Current
                    voltage = ADC_readResult(myAdc, ADC_ResultNumber_2);  // Reading Battery Voltage

                    if (voltage < vtaper)                            // MPPT, (when Vbatt lower than Vtaper)
                        {
                            mode = 1;                                // setting mode as 1 (MPPT)
                            if ( current < last_current )            // comparing Last Iout to current Iout
                                {
                                    direction = 0;                   // for monitoring purpose only
                                    control = last_control - increment;  // if last Iout larger than Iout, change direction.
                                }
                            else
                                {
                                    direction = 1;                   // for monitoring purpose only
                                    control = last_control + increment;  // if last Iout smaller than Iout, change direction.
                                }

                    // Reset Duty Cycle
                            if ( (control >= 70) || ((control <= 1)) )  // Reset Duty Cycle when out of bound
                                {
                                    control = 10;
                                }
                        }
                    else                                             // Charge Taper/Float , (when Vbatt higher than Vtaper)
                        {
                            if ( current > imin )                    // when Charging current Iout higher than Imin, execute mode 2 below
                                {
                                    mode = 2;                        // setting mode as 2 (Charge Taper)
                                    if ( voltage > (vtaper + vh) )   // compare Vbatt to Vtaper + Vh (hysteresis)
                                        {
                                            control = control - 1;   // when Vbatt higher than Vtaper + Vh, charge battery with decrement of d
                                        }
                                    else
                                        {
                                            control = control;       // when Vbatt lower than Vtaper + Vh, charge battery in constant d
                                        }
                                }
                            else                                     // when Charging current Iout less than Imin, execute mode 3 below
                                {
                                    mode = 3;                        // setting mode as 3 (Float)
                                    if ( voltage >= vfloat )         // compare Vbatt to Vfloat
                                        {
                                            control = control - 1;   // when Vbatt higher/equalt to Vfloat, charge battery with decrement of d
                                        }
                                    else
                                        {
                                            control = control;       // when Vbatt lower/equalt to Vfloat, charge battery in constant d
                                        }
                                }
                        }

            // Set PWM duty cycle
                    PWM_setCmpA(myPwm, control);                     // Set compare A value
                    PWM_setCmpAHr(myPwm, (unsigned int)(1 << 8));    //

            // Push state
                    last_control    = control;                       // Pushing States
                    last_current    = current;                       // Pushing States

            // GPIO19 Toggle
                    GPIO_setLow(myGpio, GPIO_Number_19);             // Toggle GPIO low to find out the Sampling Time.

            // Reseting counter
                    counter = 0;                                     // reset counter
            }

    // Interrupt Reset
        ADC_clearIntFlag(myAdc, ADC_IntNumber_1);                    // Clear ADCINT1 flag reinitialize for next SOC
        PIE_clearInt(myPie, PIE_GroupNumber_10);                     // Acknowledge interrupt to PIE

        return;
}
```

Figure 13. ADC ISR

The above code works well when in MPPT mode, and can also see duty cycle decrease once (***voltage***) higher than (***vtaper + vh***). However, there is a problem, the controller might bounce between mode 1 and mode 2. Reason is the ADC can also read some of the switching noise which super impose on the (rather slow) Battery Voltage.

By solving the problem mentioned above, I added a software buffer to have a clear distinguish between mode 1 and mode 2.

```
//
// Globals
//
    uint16_t                LoopCount;
    uint16_t                ConversionCount;

    uint32_t    volatile    counter             =   0;          // For extending ADC sampling time.
    uint16_t    volatile    direction           =   1;          // 1 increasing (default), 0 decreasing

    uint16_t    volatile    last_control        =   10;         // Start up duty cycle with 10.
    uint16_t    volatile    control             =   10;         // Start up duty cycle with 10.

    uint32_t    volatile    last_voltage        =   0;          // not used in MPPT
    uint32_t    volatile    voltage;                            // not used in MPPT
    uint32_t    volatile    last_current        =   0;          // last current value for comparison
    uint32_t    volatile    current;                            // Charge / output Current ADC value

    uint16_t    volatile    increment           =   1;          // increment of Duty Cycle (1/120)

    uint16_t    volatile    vtaper              =   1755;       // 13.0V    * gain ratio (135)
    uint16_t    volatile    vh                  =   50;         // 0.37V    * gain ratio (135)
    uint16_t    volatile    vfloat              =   1809;       // 13.4V    * gain ratio (135)
    uint16_t    volatile    imin                =   372;        // around 0.1A

    uint16_t    volatile    mode                =   1;          // charging mode (monitoring purpose only)
    int16_t     volatile    i                   =   1;          // between mode buffer variable
```

Figure 14. Defining Global Variables

Idea is simple, we will first define a variable (***i***), by default (***i***) should be zero, and controller would execute mode 1 MPPT. When the ADC sense the voltage higher than (vtaper), (***i***) starts increment, however, we are not jumping in mode 2 immediately, it will take consecutively 10 samples to do so, and vice versa, it also would take 10 samples decrement to jump back in mode 1 when it senses the voltage dip below (***vtaper***). Otherwise, the controller sending out Duty Cycle unchanged.

```
/* Between Mode Buffer */

   if (voltage < vtaper)                          // MPPT, (when Vbatt lower than Vtaper)
        {
            i = i - 1;                             // decrement buffer variable
            if ( i <= 0 )                          // only execute MPPT below when i smaller or equal 0
            {
                i = 0;                             // making sure i do not goes negative
                ...
                <EXECUTE MPPT HERE>
                ...
            }
        }
    else                                           // Charge Taper/Float , (when Vbatt higher than Vtaper)
        {
            i = i + 1;                             // increment buffer variable
            if ( i > 10)                           // only execute Charge Taper/Float below when i higher than 10
            {
                i = 10;                            // making sure i stays at 10
                ...
                <EXECUTE Charge Taper/Float HERE>
                ...
            }
        }
```

Figure 15. Pseudo between-mode buffer

```
//
//  adc_isr -
//
#pragma CODE_SECTION(adc_isr, "ramfuncs");
__interrupt void
adc_isr(void)
{
        counter = counter + 1;                                      // counting up

        if (counter >= 14000)                                       // change settling time here (14000 = 40mS til next ADC reading)
            {
            // GPIO19 Toggle
                    GPIO_setHigh(myGpio, GPIO_Number_19);           // Toggle GPIO high to find out the Sampling Time.

            // Reading
                    current = ADC_readResult(myAdc, ADC_ResultNumber_1);    // Reading Battery Charging Current
                    voltage = ADC_readResult(myAdc, ADC_ResultNumber_2);    // Reading Battery Voltage

                    if (voltage < vtaper)                           // MPPT, (when Vbatt lower than Vtaper)
                        {
                            i = i - 1;                              // decrement buffer variable
                            if ( i <= 0 )                           // only execute MPPT below when i smaller or equal 0
                            {
                                i = 0;                              // making sure i do not goes negative
                                mode = 1;                           // setting mode as 1 (MPPT)
                                if ( current < last_current )       // comparing Last Iout to current Iout
                                    {
                                        direction = 0;              // for monitoring purpose only
                                        control = last_control - increment; // if last Iout larger than Iout, change direction.
                                    }
                                else
                                    {
                                        direction = 1;              // for monitoring purpose only
                                        control = last_control + increment; // if last Iout smaller than Iout, change direction.
                                    }

                        // Reset Duty Cycle
                                if ( (control >= 70) || ((control <= 1)) )  // Reset Duty Cycle when out of bound
                                    {
                                        control = 10;
                                    }
                            }
                        }
                    else                                            // Charge Taper/Float , (when Vbatt higher than Vtaper)
                        {
                            i = i + 1;                              // increment buffer variable
                            if ( i > 10)                            // only execute Charge Taper/Float below when i higher than 10
                            {
                                i = 10;                             // making sure i stays at 10
                                if ( current > imin )               // when Charging current Iout higher than Imin, execute mode 2 below
                                    {
                                        mode = 2;                   // setting mode as 2 (Charge Taper)
                                        if (voltage > (vtaper + vh))    // compare Vbatt to Vtaper + Vh (hysteresis)
                                            {
                                                control = control - 1;  // when Vbatt higher than Vtaper + Vh, charge battery with decrement of d
                                            }
                                        else
                                            {
                                                control = control;  // when Vbatt lower than Vtaper + Vh, charge battery in constant d
                                            }
                                    }
                                else                                // when Charging current Iout less than Imin, execute mode 3 below
                                    {
                                        mode = 3;                   // setting mode as 3 (Float)
                                        if ( voltage >= vfloat )    // compare Vbatt to Vfloat
                                            {
                                                control = control - 1;  // when Vbatt higher/equalt to Vfloat, charge battery with decrement of d
                                            }
                                        else
                                            {
                                                control = control;  // when Vbatt lower/equalt to Vfloat, charge battery in constant d
                                            }
                                    }
                            }
                        }

            // Set PWM duty cycle
                    PWM_setCmpA(myPwm, control);                    // Set compare A value
                    PWM_setCmpAHr(myPwm, (unsigned int)(1 << 8));   //

            // Push state
                    last_control    = control;                      // Pushing States
                    last_current    = current;                      // Pushing States

            // GPIO19 Toggle
                    GPIO_setLow(myGpio, GPIO_Number_19);            // Toggle GPIO low to find out the Sampling Time.

            // Reseting counter
                    counter = 0;                                    // reset counter
            }

    // Interrupt Reset
        ADC_clearIntFlag(myAdc, ADC_IntNumber_1);                   // Clear ADCINT1 flag reinitialize for next SOC
        PIE_clearInt(myPie, PIE_GroupNumber_10);                    // Acknowledge interrupt to PIE

        return;
}
```

Figure 16. ADC ISR (w/ between-mode buffer)

- Report documents experimental data showing that the system operates in MPPT mode when the battery is sufficiently discharged (15 points)

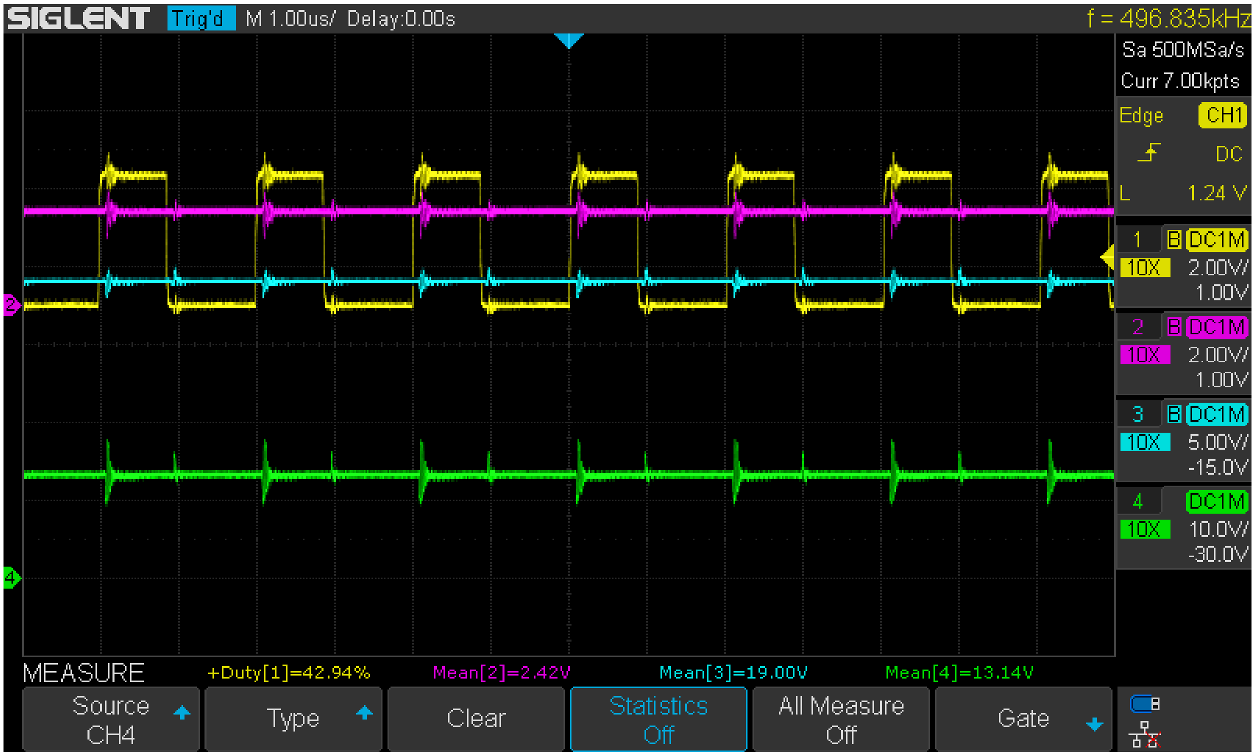


Figure 17. MPPT scope readings.

Channel 1: PWM from Launchpad.
Channel 2: Output of Current Sensor.
Channel 3: Solar Panel Voltage.
Channel 4: Battery Voltage.

Time		Panel			MCU			Battery	
		Vg (V)		Ig (A)	d	Vadc (V)	ADC value	Vbatt (V)	Icharge (A)
11:17	Sunny	17.99	16.99	0.716	0.4527	2.95	3825	13.11	0.795
11:22	Sunny	17.00	16.58	0.760	0.4650	3.10	3890	13.17	0.790
11:24	Sunny	17.06	16.55	0.740	0.4600	2.97	3830	13.17	0.780
11:25	Sunny	17.00	16.45	0.741	0.4700	3.00	3870	13.18	0.788

Table 2. MPPT sample value

Above readings was captured at 04/08/2022 11:15am Middle Village NY, Mostly Sunny at the time of measurement. The controller was running in MPPT. At the time of working, we can see Channel 3 Panel Voltage bouncing up and down slightly, Duty Cycle can be changing around 0.42-0.45 at maximum sunshine, the converter is finding the peak output. Also see video in the link below:

https://youtu.be/IQADeBF2SWQ

- Report documents experimental data showing that the system operates in charge taper mode when the battery state of charge is sufficiently high (15 points)

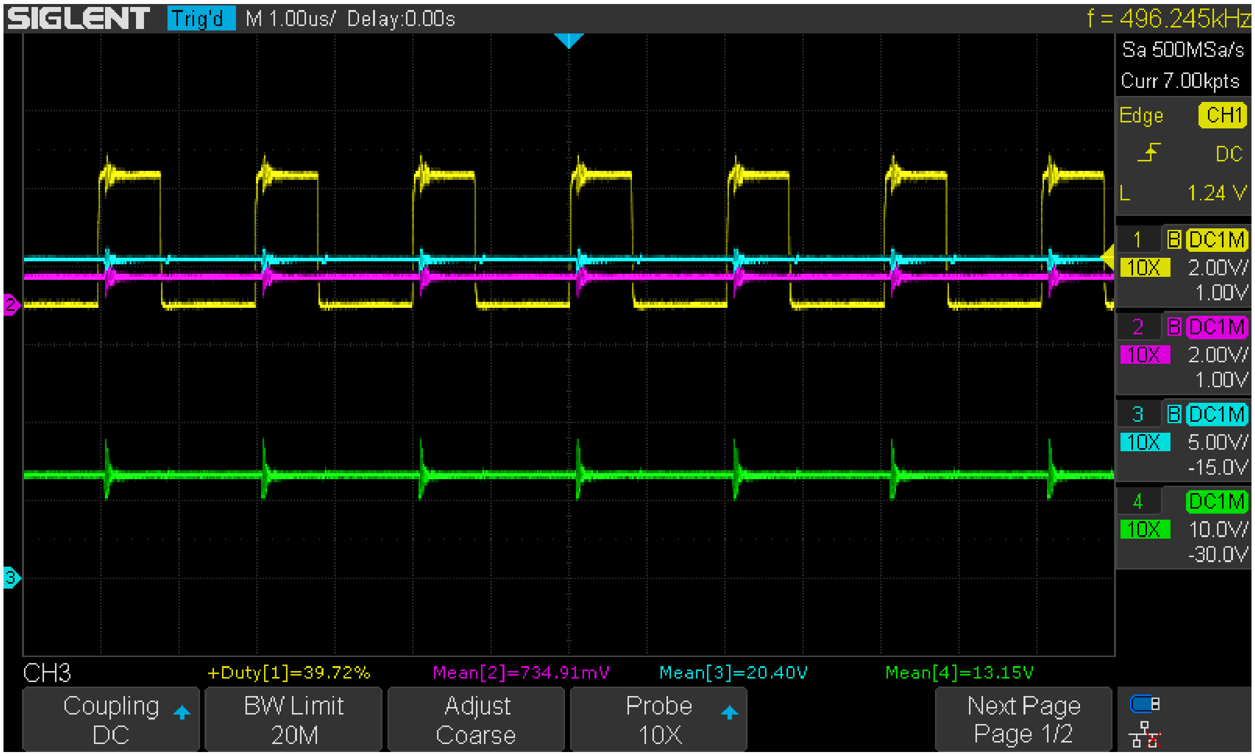


Figure 18. Charge Taper Mode scope readings.

Channel 1: PWM from Launchpad.
Channel 2: Output of Current Sensor.
Channel 3: Solar Panel Voltage.
Channel 4: Battery Voltage.

Above readings was captured at 04/08/2022 12:00pm Middle Village NY, Mostly Sunny at the time of measurement. The controller was running in Charge Taper Mode. Comparing with the previous scope reading, we can see the Duty Cycle and Charging Current decreased, and Solar Panel Voltage jump back up because less power is drawn, Battery Voltage maintaining around 13.15V. Close to what we simulated in Matlab. Also see video in the link below:

https://youtu.be/ht8GgUFioxw

(x)= Variables | Expressions | Registers

Expression	Type	Value	Address
current	unsigned long	3270	0x00000614@Data
voltage	unsigned long	1753	0x00000610@Data
control	unsigned int	50	0x00000604@Data
mode	unsigned int	1	0x0000060A@Data

Figure 19. Variables real time monitoring (MPPT)

Using the CCS debug mode, we can monitor the variable real time, above is one of the examples of Mode 1 I captured. Duty Cycle (***Control***) is running at around 50 out of 120, which is around 42% when Battery voltage (***voltage***) is below (***vtaper*** = 1755), The controller is outputting its peak power around this duty cycle.

(x)= Variables | Expressions | Registers

Expression	Type	Value	Address
current	unsigned long	2469	0x00000614@Data
voltage	unsigned long	1782	0x00000610@Data
control	unsigned int	48	0x00000604@Data
mode	unsigned int	2	0x0000060A@Data
i	int	10	0x0000060B@Data

(x)= Variables | Expressions | Registers

Expression	Type	Value	Address
current	unsigned long	1741	0x00000614@Data
voltage	unsigned long	1749	0x00000610@Data
control	unsigned int	47	0x00000604@Data
mode	unsigned int	2	0x0000060A@Data
i	int	9	0x0000060B@Data

(x)= Variables | Expressions | Registers

Expression	Type	Value	Address
current	unsigned long	1130	0x00000614@Data
voltage	unsigned long	1781	0x00000610@Data
control	unsigned int	47	0x00000604@Data
mode	unsigned int	2	0x0000060A@Data
i	int	10	0x0000060B@Data

(x)= Variables | Expressions | Registers

Expression	Type	Value	Address
current	unsigned long	1057	0x00000614@Data
voltage	unsigned long	1756	0x00000610@Data
control	unsigned int	47	0x00000604@Data
mode	unsigned int	2	0x0000060A@Data
i	int	9	0x0000060B@Data

Figure 20. Variables real time monitoring (Charge Taper Mode)

Above values are captured around 10-15 minutes apart, in Charge Taper Mode (Mode 2), we can see the duty cycle decrement from 50 down to 47 slowly, Charging current is obviously decreasing with constant voltage charging. By the time of experiment the current decreases continuously while maintaining around 13.3V Battery Voltage, it took quite some time until it will finally reach Float Mode (Mode 3).

Measurement Equipment:

- FLUKE 289 TRUE RMS MULTIMETER (Calibrated by TRANSCAT on 06/15/2021)
- SIGLENT SDS 1104X-E DIGITAL STORAGE OSCILLOSCOPE
- EXTECH 1430 TRUE RMS MULTIMETER

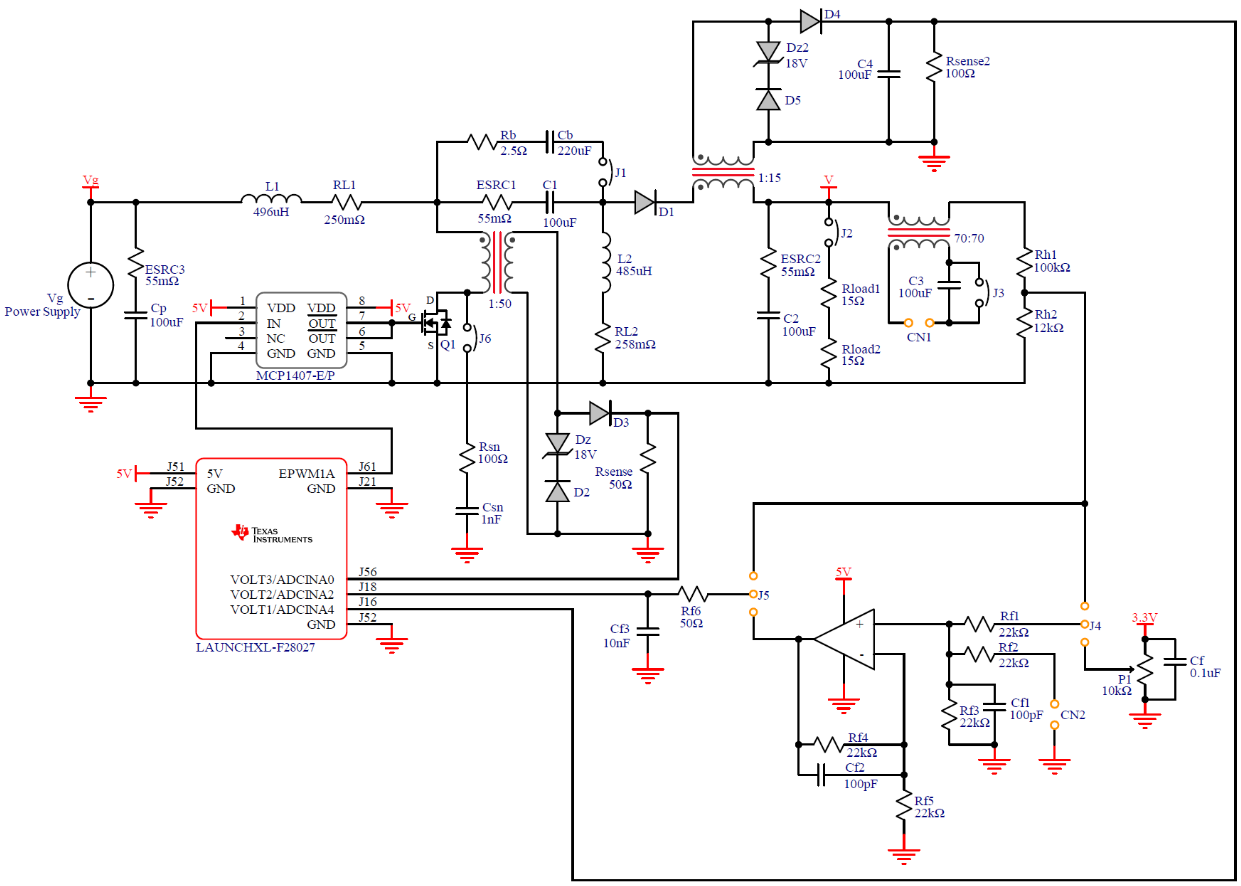
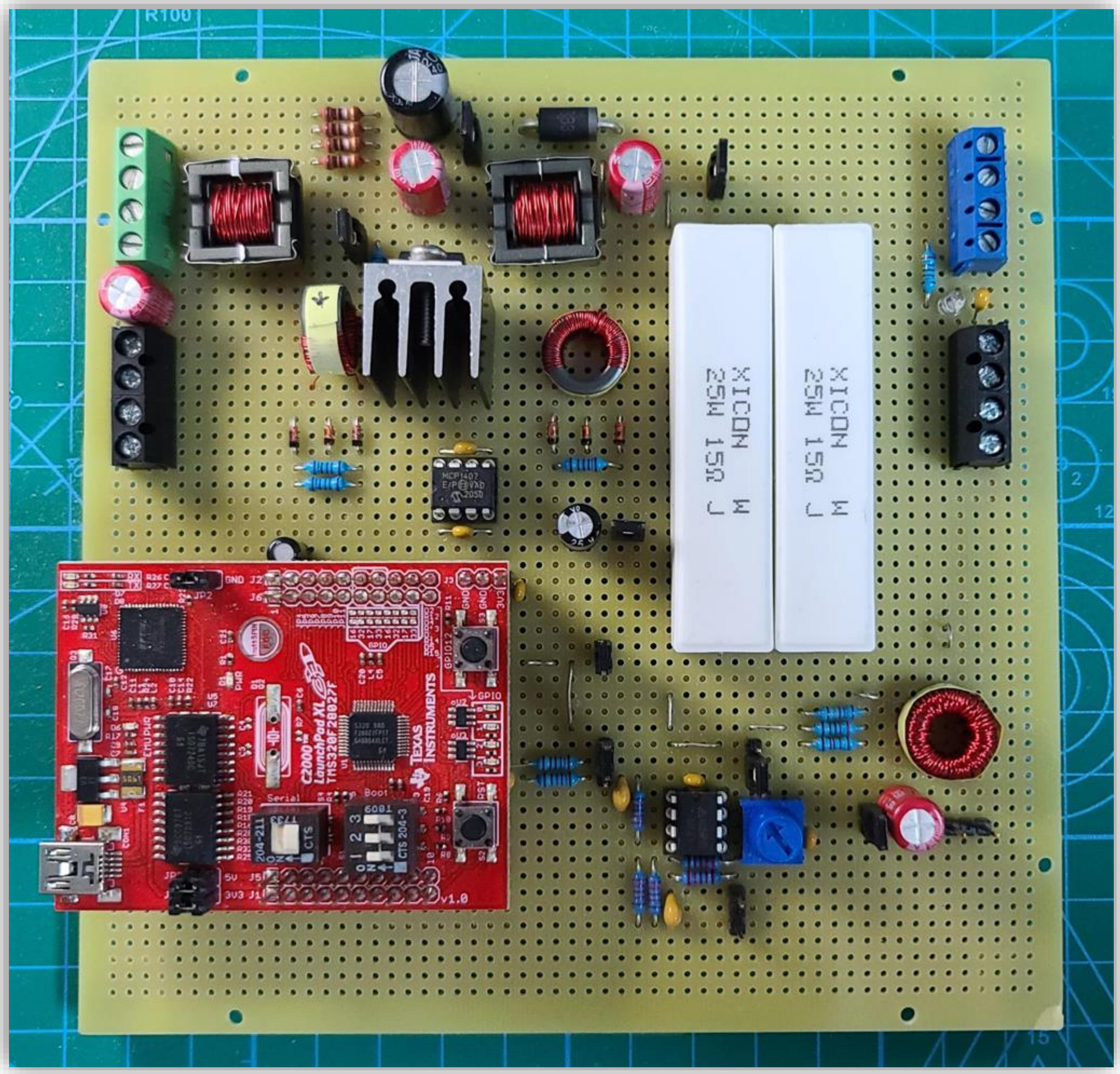
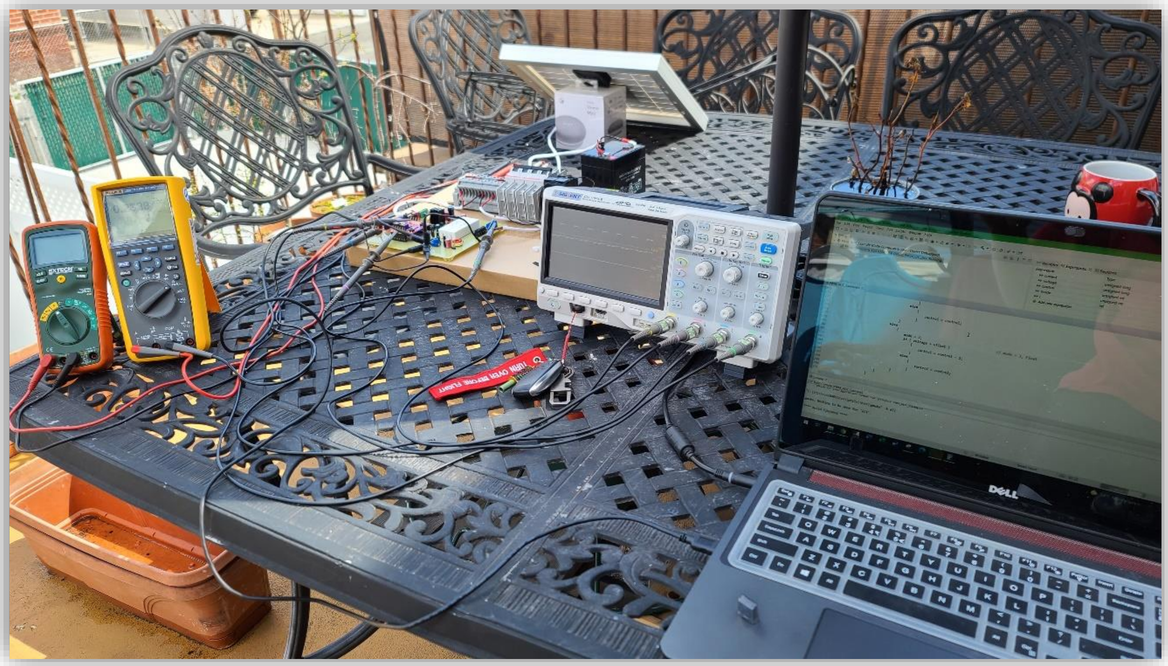


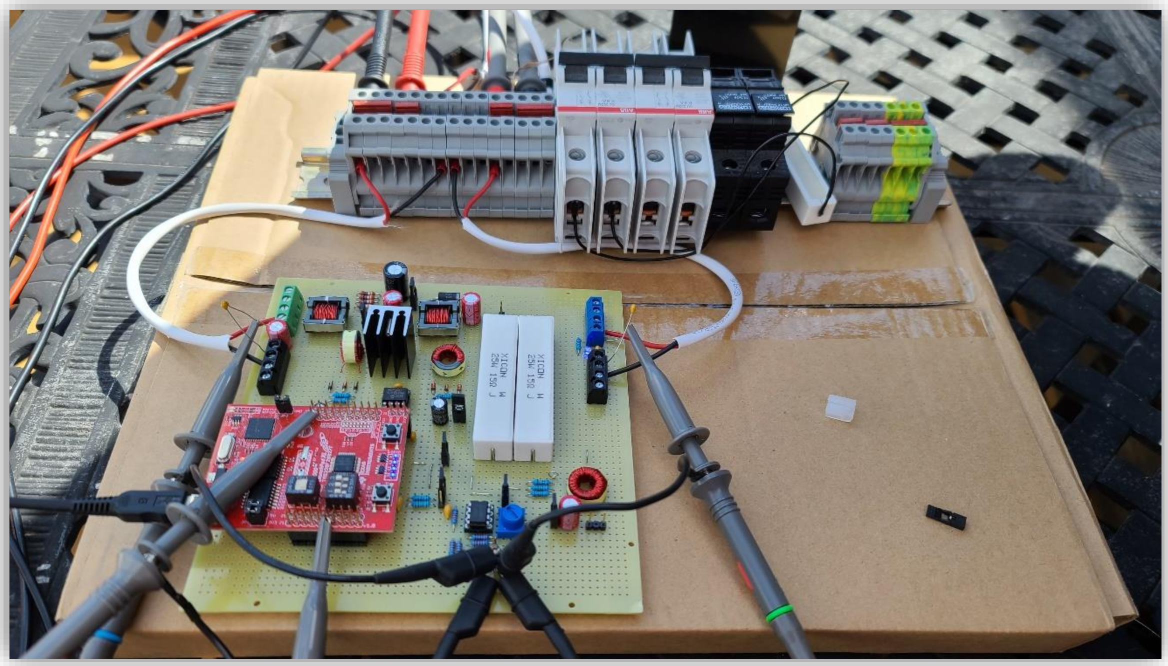
Figure 21. Complete Schematic



Picture 1. Complete Circuit with Current Sensing Circuit

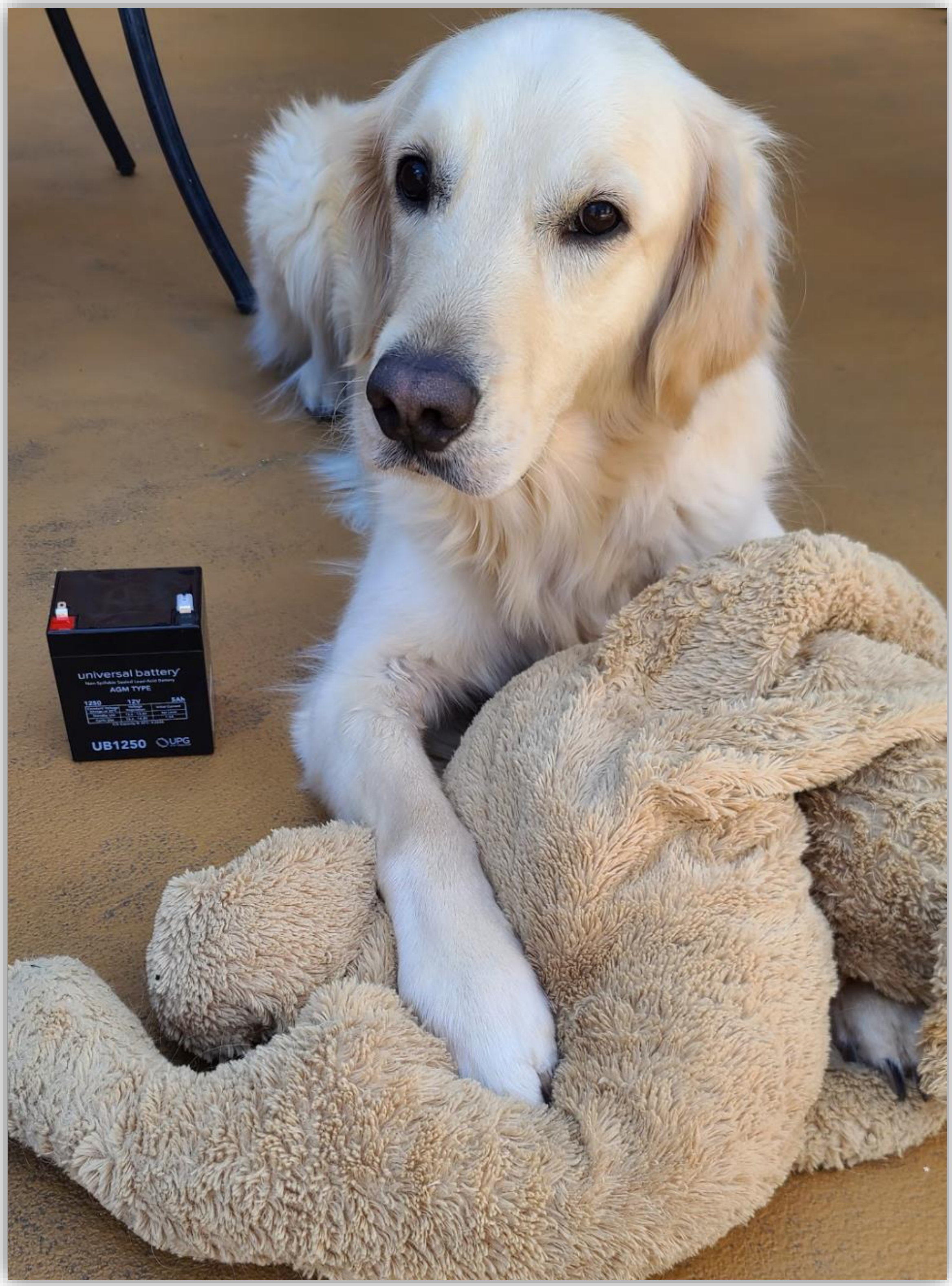


Picture 2. Testing Set up 04/08/2022 12:30pm Middel Village, NY (mostly Sunny)



Picture 3. Testing Set up 04/08/2022 12:30pm Middel Village, NY (mostly Sunny)

[1] Battery - UB1250
https://d18ky98rnyall9.cloudfront.net/y1j3f1z1TLaY939c9dy2rw_fff75a54e01c4ebe931f3da8838083b9_D5741.pdf?Expires=1649116800&Signature=ARgE83ldS4C5TSi2xt5UpGJxe~x7Pj-~2Z9pzpKBlOMmpzfv19v7KvyPArVhUMFsVDaJHRo7Djncjwz7IkXHXyLP8psXhSZ6d5IAOprSId46Hi~IRf8hZAoOsXKoRJcaCpb9d~xiSNM8poYLnoIFNd~G9DvDt5bq~v4ZD8ZhZ48_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A



Picture 4. My project assistant, “Donut” - The Golden Retriever.