

LAB 4 – MAXIMUM POWER POINT TRACKING FOR PV SYSTEM USING PERTURB AND OBSERVE (P&O) TECHNIQUE

Group Information

Name	Student ID	Name	Student ID

Report Submission

Each group only requires to submit one report. Waveforms, calculations and answers should be laid out neatly and correctly. Marks will be lost for missing and illegible work. Submit your report via UTSOnline. Note the deadline of submission on UTSOnline.

Declaration of Originality

The work contained in this assignment, other than that specifically attributed to another source, is that of the author(s). It is recognised that, should this declaration be found to be false, disciplinary action could be taken and the assignments of all students involved will be given zero marks. In the statement below, I have indicated the extent to which I have collaborated with other students, whom I have named.

Statement of Collaboration and Signatures

--

Marks

Pre-lab work	
Lab work	
Total marks	

2.3 Working principle of P&O

The working principle of the P&O MPPT is explained as follows: In Figure 1, the PV panel voltage V_{dc} is sensed and scaled via the voltage divider (R_{in1} and R_{in2}) to an acceptable voltage range for the Arduino (0-5V which is corresponding to 0-1023 after the ADC). The RC filter formed by R_1 and C_1 is to filter any switching noise from the synchronous buck converter to pin A0 (ADC) of the Arduino. If not properly filtered, the noise may affect the operation of the Arduino (e.g. false reset of the Arduino program, malfunction, etc). Next, the PV panel current, I_{PV} , is obtained via the output voltage V_{out} , as we can relate the input current to the output voltage by the following equation:

$$P_{PV} = I_{PV} \times V_{PV} = DI_{out} \times V_{PV} = \frac{DV_{out}}{R_{out}} \times V_{PV} \quad (1)$$

The output voltage is sensed and scaled via the voltage divider (R_{out1} and R_{out2}). Another RC filter formed by R_2 and C_2 provides a smoother signal to pin A1 (ADC pin) of Arduino. According to Equation (1), this signal at pin A1, together with the PV panel voltage at pin A0, carries the PV panel current information, I_{PV} .

Once the power information is obtained we can start working on the algorithm. Let us assume the operation point of the converter starts at point A as shown in Figure 2. The power at point A is stored. Now the duty cycle decreases (**Perturb**) so the operation point moves to point B. The power at point B is stored. After comparing the powers at A and B (**Observe**), the algorithm decides to keep decreasing the duty cycle as it should continue to move to the right of the P-V curve. Now it arrives at point C then point D. When comparing the powers at points C and D, the algorithm will decide to keep decreasing the duty cycle. Soon it reaches at point E. However, the power at point E is smaller than that of point D. Therefore the algorithm will increase the duty cycle instead of decreasing it. The operating point now goes back to point D. After the comparison, the algorithm would decide to keep increasing the duty cycle (to move to the left of the P-V curve) as power at point D is greater than that of point E. The next operating point of this action will be at point C. Hence it can be observed that now the MPPT will be moving among points C, D and E, in a back and forth manner continuously.

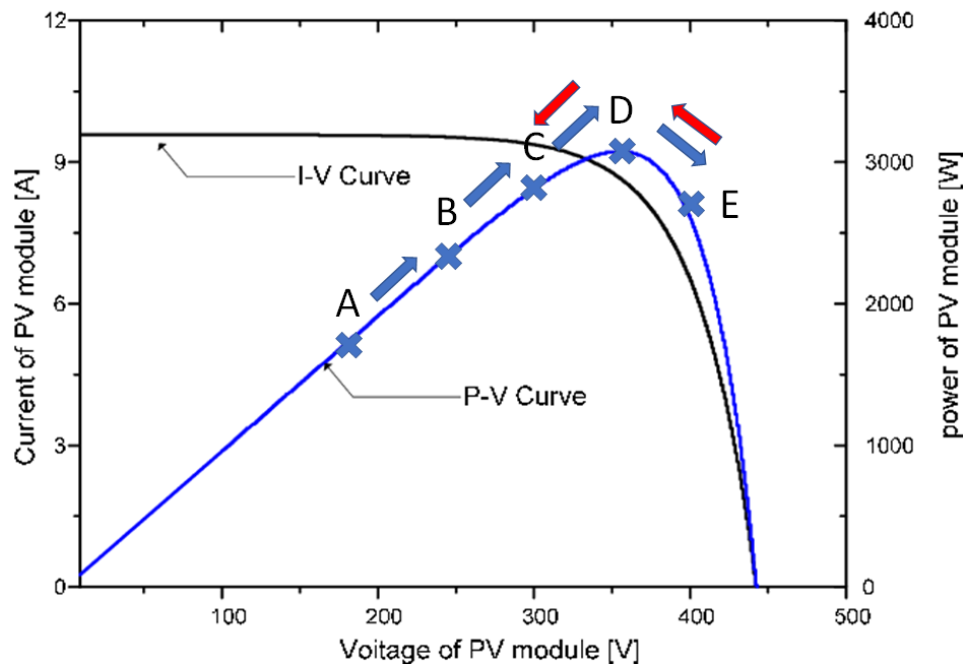


Figure 2: P&O algorithm stores the power information and changes the buck converter's operating point to obtain another power information (Perturb). After the comparison between the two powers (Observe), it decides the direction of movement via changing the duty cycle (i.e. decrement or increment). This process is continuous.

Note that Equation (1) assumes the buck converter operates in a lossless condition. To include the power loss, we may insert a conversion efficiency term η to the equation. However, since the P&O MPPT algorithm only concerns about the relative values as discussed in the previous paragraph, the efficiency and other constant values can be omitted. In our implementation, we can just retain the time varying variables and simplify the equation to

$$\text{Power information at point X} = V_{out}^2 \times V_{PV}^2. \quad (2)$$

To further simplify the coding and computational time of the Arduino, Equation (2) can be further reduced to

$$\text{Power information at point X} = V_{out} \times V_{PV}. \quad (3)$$

3 Pre-lab Work

- i. Study Section 2 and explain in your own words why the power information at a particular operation point can be simplified as that in Equation (3).
- ii. Design, using pseudo code, a flow chart or your own descriptive language, a P&O algorithm for the buck converter to achieve MPPT.

4 Equipment and Components

- 1 × Assembled testing board (Synchronous Buck Converter)
- 1 × Breadboard
- 2 × Digital multimeter (benchtop and/or hand-held)
- 1 × Laptop computer with Arduino IDE installed or Web Editor Account created (each group to prepare)
- 1 × Mini-B USB download cable
- 1 × Arduino Nano board
- 1 × Digital Storage Oscilloscope (DSO)
- 2 × 10:1 Oscilloscope Voltage Probe
- 1 × Two channel Power Supply up to 30V 1A
- 1 × Current Probe with amplifier
- 1 × PV emulator board (mainly diodes and connector sockets on a PCB)
- 1 × Variable resistor bank (box)
- 1 × Capacitor (100uF/50V) for the output of the buck converter
- 2 × 1.8 k Ω for R_{in1}
- 2 × 8.2 k Ω for R_{in2}
- 2 × 1 k Ω for R_1
- 2 × 10 nF for C_1

5 Experiments

5.1 PV emulator

In the first experiment, you are going to plot the I-V and P-V curves of the PV emulator. Even you have done a similar test in Lab 3, the power supply setting and diode string board will be different. You will need a new set of measurement data to compare the performance of the P&O algorithm.

- a) Turn on the power supply. Set first output of the power supply to **0 V** with the current limit at **0A**.
- b) Turn off the power supply.
- c) Connect the power supply to the PV emulator board.
- d) Connect the resistor box to another terminals of the PV emulator board via a digital meter for current measurement. Set the resistance value to open-circuit (**O/C**).
- e) Connect another digital meter for voltage measurement of the diode string.

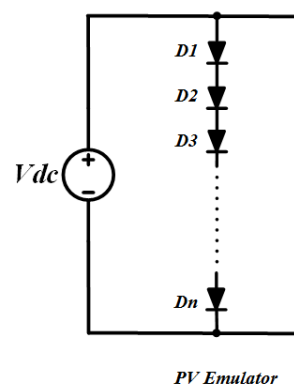


Figure 3: Circuit diagram of a PV emulator.

- f) Turn on the power supply. Turn the voltage knob to around 20V-21V while keeping the current-limit to **0A**. Stop turning the voltage knob when you see the current-limit LED is turned on. Now turn the current-limit knob from 0A to around **0.4A**. Make sure the current-limit LED is kept on. If the current-limit LED is off, increase the supply voltage slightly until it turns on again. This completes the power supply setting procedure.
- g) Note that the diodes will grow **HOT** when the power supply is turned on. Please **DO NOT TOUCH THE DIODES when power supply is ON**.
- h) Vary the resistance values from open circuit (OC) condition, to $270\ \Omega$ and so on down to $0\ \Omega$. Record the current and voltage readings for each resistance value.
- i) Use a **spreadsheet program** to plot the I-V and P-V curves to locate the maximum power point (MPP). From the graphs, identify the corresponding voltage at MPP. This becomes the reference voltage for the constant voltage reference MPPT.
- j) Repeat steps c) to h) using other current-limit values such as 0.3A and 0.2A while not changing the voltage knob.
- k) Once finished, turn off the power supply. Remove the resistor box connections to the PV emulator. The resistor box will be used for other purpose (i.e. buck converter output) in later section.

5.2 Digital PWM Generation

- a) Connect the USB cable between the computer and the Arduino Nano board. Run Arduino IDE or equivalent editor program. (**Warning: DO NOT connect the benchtop power supply 21V to the assembled testing board at this stage yet.**)
- b) Upload an empty file to the Arduino first to make sure the communication between Arduino and the laptop is proper.
- c) Download the **PWM_Master.zip** file available in UTSONline subject laboratory exercises folder into the same place where you saved your current program.
- d) Click on the **Sketch** tab then the **Include Library** tabs, select **Add .ZIP Library** from the drop-down menu and click.
- e) A dialog box will pop up. On the File of type field, select the **PWM_Master.zip** to be included in the programming editor.
- f) Download the **PWM.h** file available in UTSONline subject laboratory exercises folder into the same place where you saved your current program.

- g) Copy the sample code, and program the Arduino to produce PWM pulses at 100kHz and 50% duty cycle according to your pre-lab work.
- h) Observe and record the PWM signal from the PWM pin (D10) of the Arduino board using the DSO.

5.3 Setting up the step-down synchronous buck converter

- a) Connect the power supply 21V (power supply is off) via a digital meter (for current measurement) and 0V pins/sockets to the board **without connecting the PV emulator**. Refer to Appendix A for details.
- b) Turn off the power supply.
- c) Check if jumper J101 of the assembled testing board (synchronous buck converter) is in place.
- d) Connect the 100 μ F capacitor and the resistor bank (select 33 Ω) to the output (HB1 Vind) and ground terminals of the assembled testing board. Refer to Appendix A for details. Use a breadboard for this step if necessary.
- e) Connect the ground pin of the probe to the ground of the testing board. Refer to Appendix A for details.
- f) Connect PWM and 0V from the breadboard to pins 1 and 2 (positive, i.e. shorting pins 1 and 2) and pin 6 (negative) on J102 respectively of the assembled testing board.
- g) Connect the positive terminal of the PV emulator to pin 5 or 6 on J100 (or J103) and negative terminal to pin 1 or 2 on J100 (or J103) of the assembled testing board. Refer to Appendix A for details.

5.4 Testing of the buck converter

- a) After checking the connections, turn on the power supply.
- b) Observe the PWM signal (using as main triggering signal on DSO).
- c) Use other channels to observe the mid-point (TP102), inductor current, and output voltage. Record the waveforms of all four channels including the PWM signal.

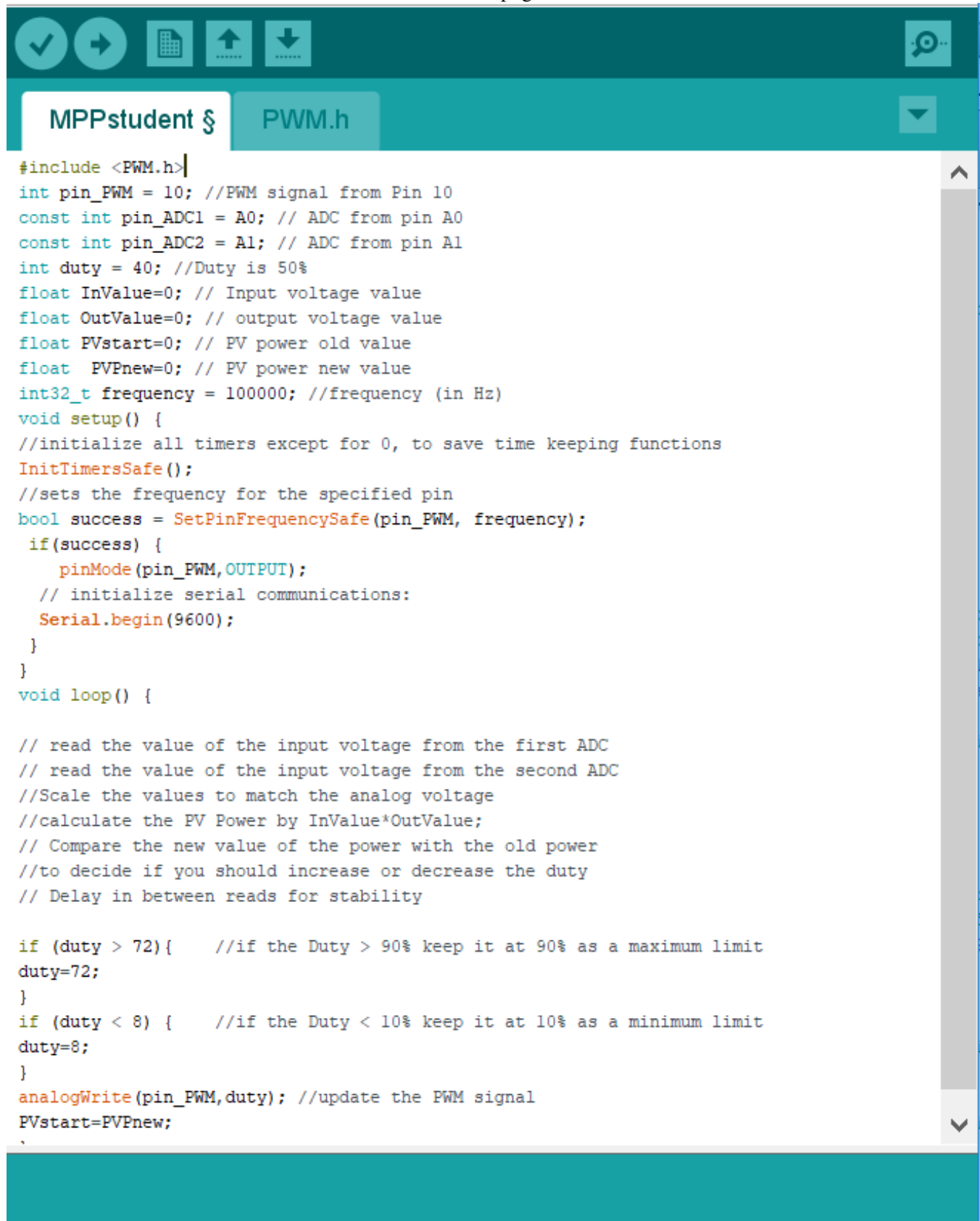
5.5 Programming P&O MPPT algorithm

- a) Input the following code to your Arduino programming editor as shown in Fig. 4.
- b) Based on part ii) of your prelab work. Enter the missing part of the code to complete the Perturb and Observe (P&O) MPPT algorithm.
- c) Download the code to the Arduino.

5.6 Testing of P&O MPPT

- a) Make sure power supply is turned off.
- b) Based on the circuit shown in Figure 1, build the input voltage and output voltage divider circuits on a breadboard, then reconnect PV emulator between the 21V power supply and the buck converter.
- c) Turn on the power supply.

code.png



```

#include <PWM.h>
int pin_PWM = 10; //PWM signal from Pin 10
const int pin_ADC1 = A0; // ADC from pin A0
const int pin_ADC2 = A1; // ADC from pin A1
int duty = 40; //Duty is 50%
float InValue=0; // Input voltage value
float OutValue=0; // output voltage value
float PVstart=0; // PV power old value
float PVPnew=0; // PV power new value
int32_t frequency = 100000; //frequency (in Hz)
void setup() {
//initialize all timers except for 0, to save time keeping functions
InitTimersSafe();
//sets the frequency for the specified pin
bool success = SetPinFrequencySafe(pin_PWM, frequency);
if(success) {
pinMode(pin_PWM,OUTPUT);
// initialize serial communications:
Serial.begin(9600);
}
}
void loop() {

// read the value of the input voltage from the first ADC
// read the value of the input voltage from the second ADC
//Scale the values to match the analog voltage
//calculate the PV Power by InValue*OutValue;
// Compare the new value of the power with the old power
//to decide if you should increase or decrease the duty
// Delay in between reads for stability

if (duty > 72){ //if the Duty > 90% keep it at 90% as a maximum limit
duty=72;
}
if (duty < 8) { //if the Duty < 10% keep it at 10% as a minimum limit
duty=8;
}
analogWrite(pin_PWM,duty); //update the PWM signal
PVstart=PVPnew;
}

```

Figure 4: Arduino code for P&O MPPT implementation.

- d) Observe on the DSO the PWM signal, input current to the buck converter (i.e. clamp the wire from 21V) and input voltage to the buck converter.
- e) Use the MATH function to obtain the input power (i.e. product of input voltage and input current). Record all waveforms.
- f) Now vary the current limit on the 21V power supply between 0.3A and 0.5A (get a few sets e.g. 0.35A, 0.4A, 0.45A, etc.). **Don't let supply voltage to go below 18V as the buck converter will stop working.** Observe that if the input power would increase when power supply current increases and vice versa. If it happens, your P&O MPPT algorithm is working.
- g) Capture the waveforms on the DSO. Record the average values of input voltage, input current and duty cycle of the PWM for different current limit values.

5.7 Analysis

- i) Explain your part of the code to achieve P&O MPPT with the support of experimental results.
- ii) Compare the P&O MPPT performance in Section 5.6 against the identified true MPPs in Section 5.1.
- iii) Is P&O a perfect maximum power point algorithm? Why or why not? Explain your argument with the experimental results if possible.
- iv) Suggest a way to improve the tracking accuracy of P&O.

Appendix

- (1) Top view of the Assembled Testing Board PCB
- (2) Schematic diagram of the Assembled testing Board

