LAB 3 – MAXIMUM POWER POINT TRACKING FOR PV SYSTEM USING CONSTANT VOLTAGE REFERENCE

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	

1 Purpose

This lab exercise builds upon the understanding of PV panel through Lab 1 and power conversion through Lab 2 and develops a simple maximum power point tracking (MPPT) for PV system using the same buck converter in Lab 2. In this lab, students will program the Arduino microcontroller to implement a constant voltage reference MPPT method and test it on a PV emulator.

2 Introduction

As we have learned in Lab 1 that the PV cell has a non-linear electrical characteristic, there is only one maximum power point (MPP) for a given solar intensity level. However this MPP changes when the solar intensity level changes. A fixed resistor simply connected to the PV terminals will not be able to achieve a continuous maximum power point tracking performance. The power converter however is in fact a variable resistor through pulse-width modulation (PWM). The change of duty cycle of the power transistor varies the equivalent resistance seen by the PV panel, hence tracking the MPP continuously.

Fig. 1 shows a circuit diagram of the main connections for this laboratory exercise. Instead of using an actual PV panel, you are going to use a PV emulator which consists of a benchtop DC power supply working in current-limit mode and a diode string connected in parallel with the DC power supply. The buck converter is the same converter circuit in Lab 2.

Constant voltage reference MPPT is an approximate MPPT method. As shown in Fig. 2, a fixed resistor can only provide one single MPP for a given range of solar intensity levels. However, for constant voltage reference MPPT method using a power converter, the operating voltage points will be always be fixed due to the feedback (feedforward more precisly) regulation of the PV cell/panel terminal voltage through the voltage divider and microcontroller. Although, similar to the fixed resistor approach, only one true MPP can be tracked, the other operating points under different solar intensity levels are much closer to the actual MPPs. This method is often used due to its simplicity of implementation as only PV voltage is sensed. Other more advanced methods such as Perturb and Observe, as we will work on in Lab 4, requires sensing of both PV voltage and current. This adds complexity and reliability issue in the implementation.

The working principle of the constant voltage reference MPPT is briefly explained as follows: 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). The RC filter formed by R_1 and C_1 is to filter any switching noise from the synchronous buck converter to pin any ADC pin which may affect the operation of the Arduino. An internal reference voltage is preset in Arduino and compared with the reading of the ADC. Corresponding action, namely, changing of duty cycle, will be taken place in a continuous manner until the value at ADC is equal to the internal reference voltage.

3 Pre-lab Work

i) Explain the working principle of the PV emulator. You can simulate the PV emulator circuit in LTSpice to understand how it works. The number of diodes is around 25 to 30. Instead of a DC voltage source, choose to use a DC **current** source in the component library. The operating range of the current source is between 0.3A and 0.5A. To plot the I-V and P-V curves on LTSpice, please watch the YouTube video entitled, "Plotting electrical characteristics of PV cell using LTSpice" on UTSOnline. It is placed under the Labs and Project folder. [Hint: Use Ampere's Law to explain different currents flowing in the circuit from short circuit to open circuit conditions.]

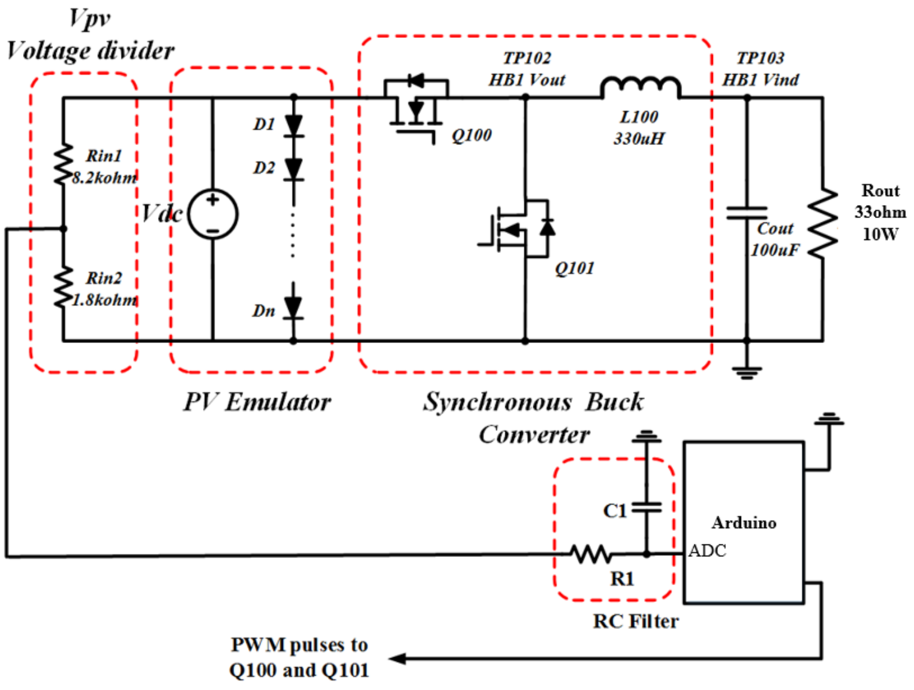


Figure 1: Main connections of the experiment for Lab 3.

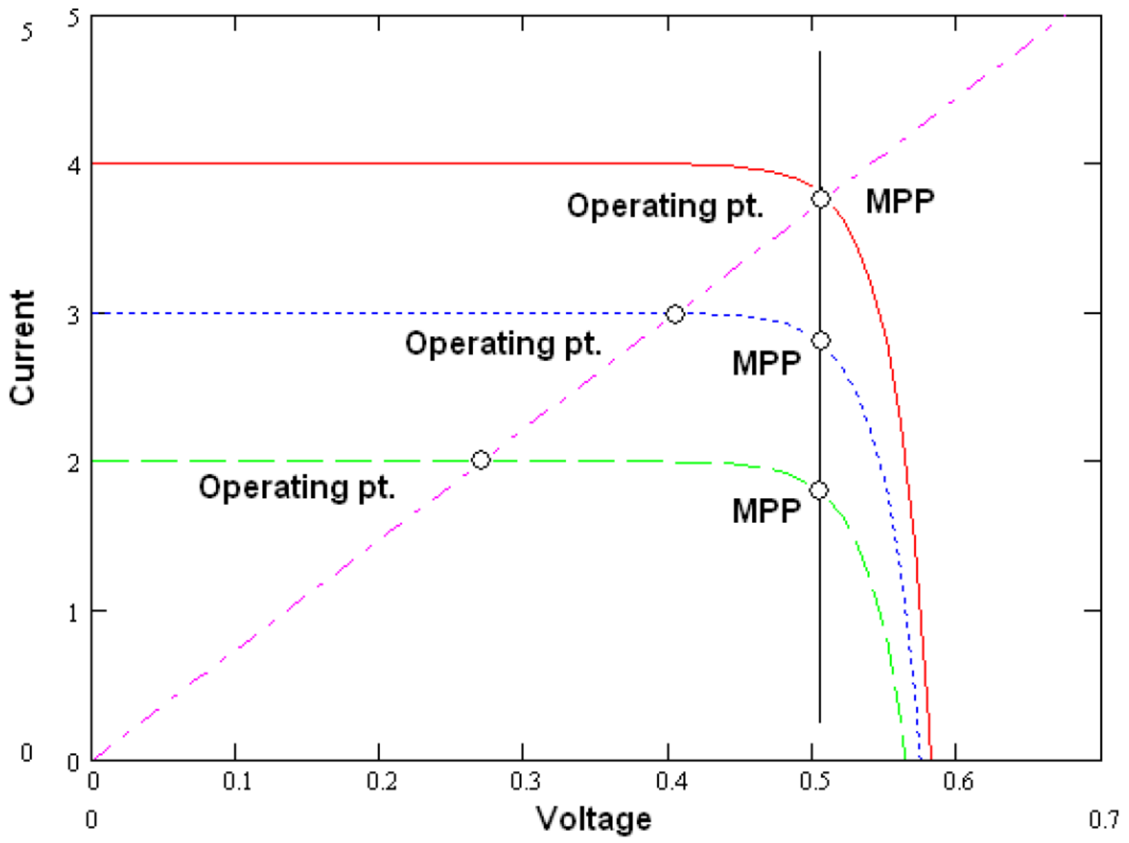


Figure 2: Comparison of fixed resistance and constant voltage reference MPPT on a single PV cell. Same principle applies to PV panel as it is a series connection of many PV cells together.

ii) Explain how the duty cycle should change to make the PV panel voltage constant and operate at the reference voltage. For a given insolation level the PV panel voltage is not yet regulated but is located on the right hand side of the I-V curve that is away from the MPP (i.e. Point A) as shown in Fig. 2, should the duty cycle decrease or increase in order to move toward the MPP? How about when the operating point is on the left-hand side of the I-V curve further away from the MPP (i.e. Point B)?

4 Program an Arduino Nano

4.1 What is Arduino?

Arduino is an inexpensive microcontroller based electronic board. It has easy-to-use hardware and software platforms and both of them are open-source and extensible. It uses Arduino programming language that is based on Java, the reference can be found at https://www.arduino.cc/reference/en/.

4.2 Programming Editors

Arduino IDE is an cross-platform development tool for Aruidno boards. It supports Windows, Mac and Linux operating systems. **Arduino Web Editor** is an alternative programming tool. It allows users to upload their sketches to the cloud and code online. Both the Arduino desktop IDE and the Web Editor can be found from https://www.arduino.cc/en/Main/Software. In this lab manual, **Arduino IDE** is used and explained. A detailed guide has been provided on the website called **Getting Started with Arduino Web Editor on Various Platforms** in helping using the Arduino Web Editor.

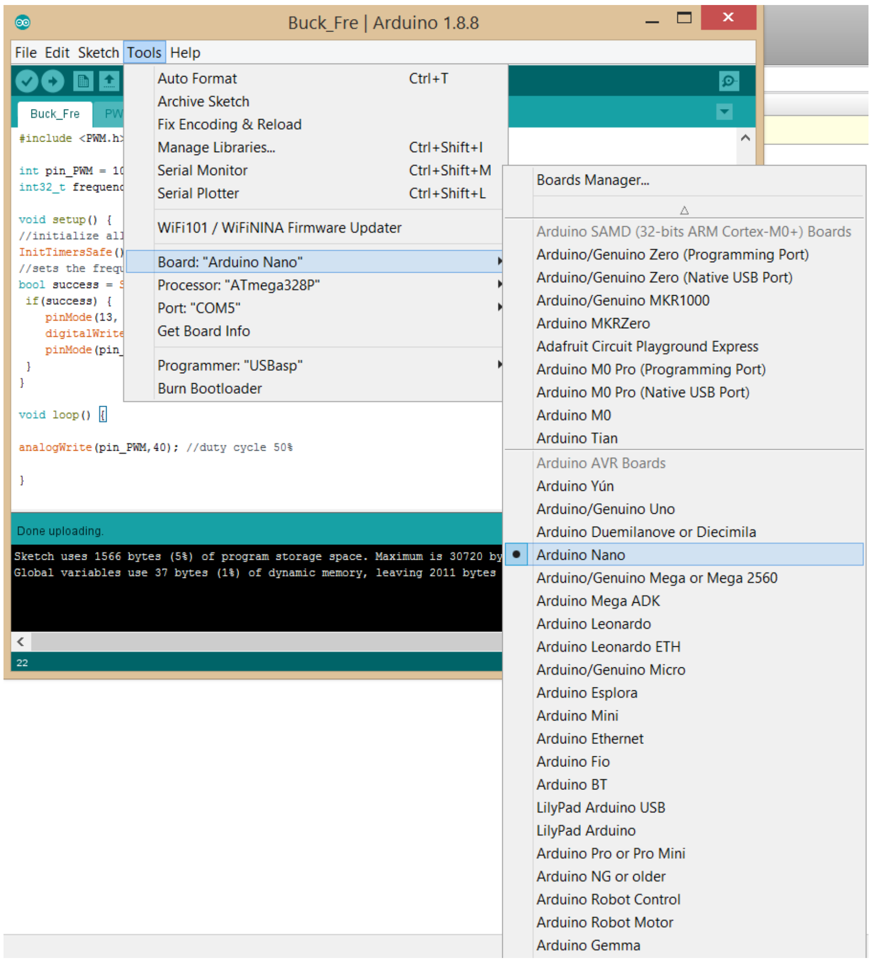
4.3 Board, Processor and Port Selection

After installed the Arduino IDE, run the program. The specific Arduino board, processor and port will need to be selected, as shown in Fig. 3.

- Click on the **Tools** tab then the **Board** tabs, select **Arduino Nano** from the drop-down menu and click.
- Click on the **Tools** tab then the **Processor** tabs, select **ATmega328P** from the drop-down menu and click.
- Click on the **Tools** tab then the **Port** tabs, select the availabled **Com** from the drop-down menu and click. Note that this step will be completed when the Arduino board is connected.

4.4 Programming Arduino Nano

Arduino Nano is a compact, breadboard-friendly board. The top and bottom view of the board are shown in Figs. 4 and 5 respectively. More details can be found from https://store.arduino.cc/usa/arduino-nano. The Nano has 6 PWM output pins, which are pin 3, 5, 6, 9, 10 and 11. They provide 8-bit PWM output capability using the command called **analogWrite**. They have different default frequencies and are controlled by the timers 0, 1 and 2. The frequencies cannot be changed by users if using one simple function/command provided by the Arduino programmming language. An approach to solve this problem is to include a PWM library.



Board: "Arduino Nano"
Processor: "ATmega328P"
Port: "COM5"
Programmer: "USBasp"

```
#include <PWM.h>

int pin_PWM = 10
int32_t frequenc

void setup() {
//initialize all
InitTimersSafe()
//sets the frequ
bool success = S
 if(success) {
    pinMode(13,
    digitalWrite
    pinMode(pin_
 }
}

void loop() {

analogWrite(pin_PWM,40); //duty cycle 50%

}
```

```
Done uploading.
Sketch uses 1566 bytes (5%) of program storage space. Maximum is 30720 by
Global variables use 37 bytes (1%) of dynamic memory, leaving 2011 bytes
```

Figure 3: Arduino IDE, choose Arduino board, processor, and port.

The PWM library facilitates users to change the PWM frequencies as desired. In the meantime the **analogWrite** function is no longer defaulted by the 8-bit resolution but it becomes a period dependent value. The period of frequency and analog duty cycle can be determined by equations (1) and (2). The system clock frequency f_{clk} of the ATmega328P is 16MHz, the prescaler N is 1, and d is the duty cycle. A sample code can be found in the Laboratory Exercises folder, and the code is shown in Fig. 6. A 100kHz frequency is set for Pin 10. And based on equation (1), you should be able to calculate the period counter (OCR1x where x = A is for Pin 9 and x = B is for Pin 10) of the frequency. The duty cycle value for **analogWrite** can be calculated based on equation (2).

Notice: In this lab, you will only need to change the frequency value in line **int32 _ t frequency = [value];** and the analogWrite duty cycle value in line **analogWrite(pin _ PWM, [value])**. You can use either Pin 9 or Pin 10, as in the sample code, we have only set timer 1 (i.e. OCR1A for Pin 9, OCR1B for Pin 10) which controls Pins 9 and 10 PWM outputs; they share the same frequency. Otherwise you will need to set timer 0 or timer 2 for other PWM output pins.

$$\text{Frequency (Hz)} = \frac{f_{clk}}{2 \times N \times (1 + \text{OCR1x})} \tag{1}$$

$$\textbf{AnalogWrite}\ \text{Duty Cycle} = (1 + \text{OCR1x}) \times d \tag{2}$$

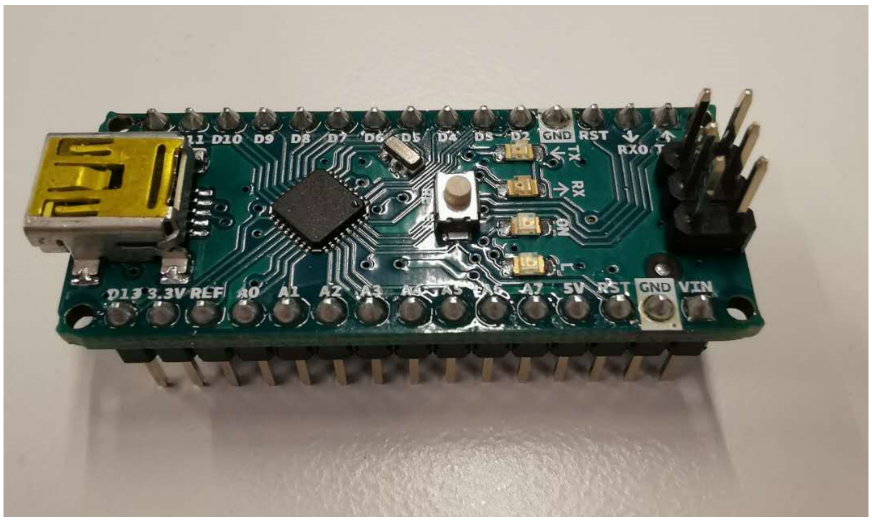

D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX
TX RX ON L
D13 3.3V REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN

Figure 4: Arduino Nano top view.

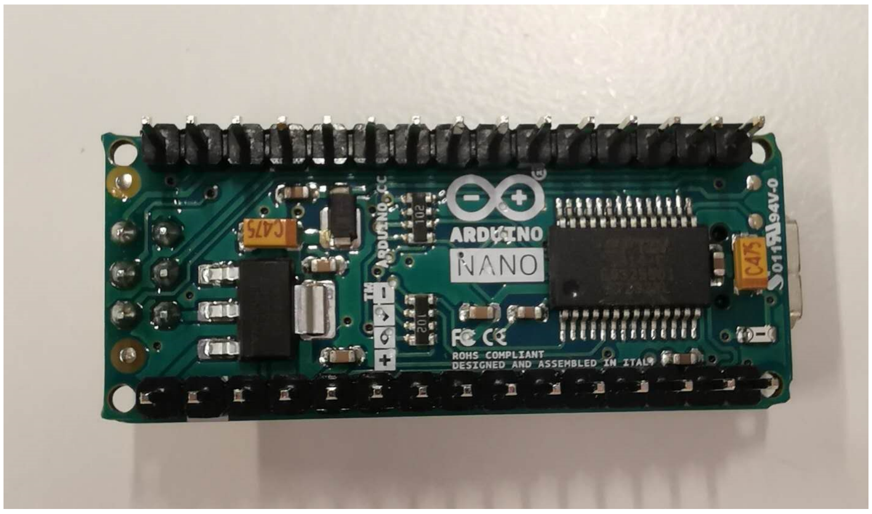
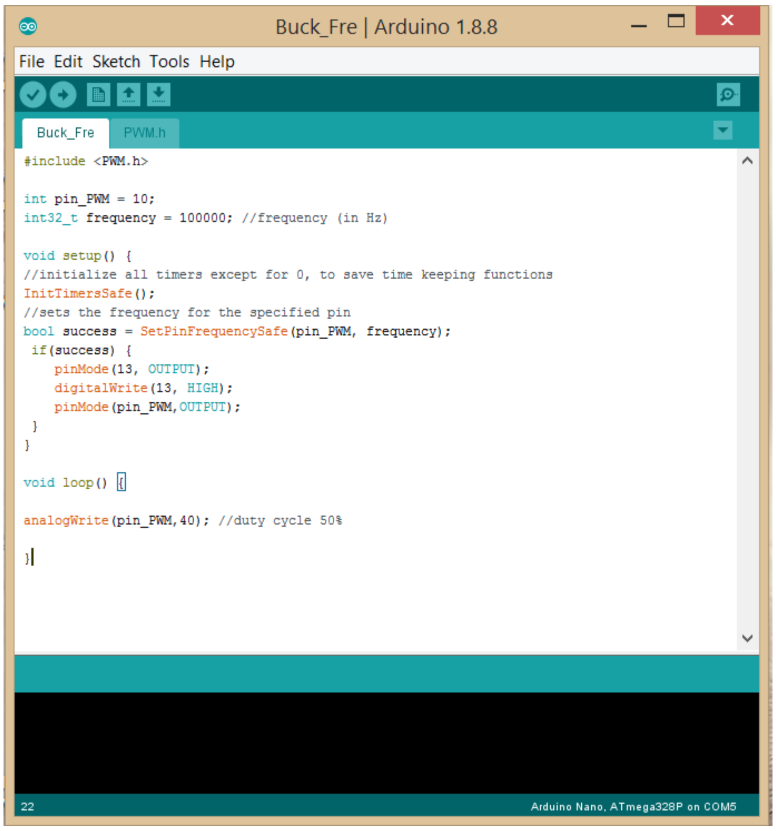


Figure 5: Arduino Nano bottom view.



```
#include <PWM.h>

int pin_PWM = 10;
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(13, OUTPUT);
    digitalWrite(13, HIGH);
    pinMode(pin_PWM,OUTPUT);
 }
}

void loop() {

analogWrite(pin_PWM,40); //duty cycle 50%

}
```

Figure 6: Arduino sample code to generate a PWM signal via pin 10 with 100kHz frequency and 50% duty cycle.

Parameters	Values
Input voltage V_{in}	24V
Inductor L	330μH
Output capacitor C	100µF
Duty cycle	50%
Switching frequency	100kHz
Output resistor R_L	33Ω

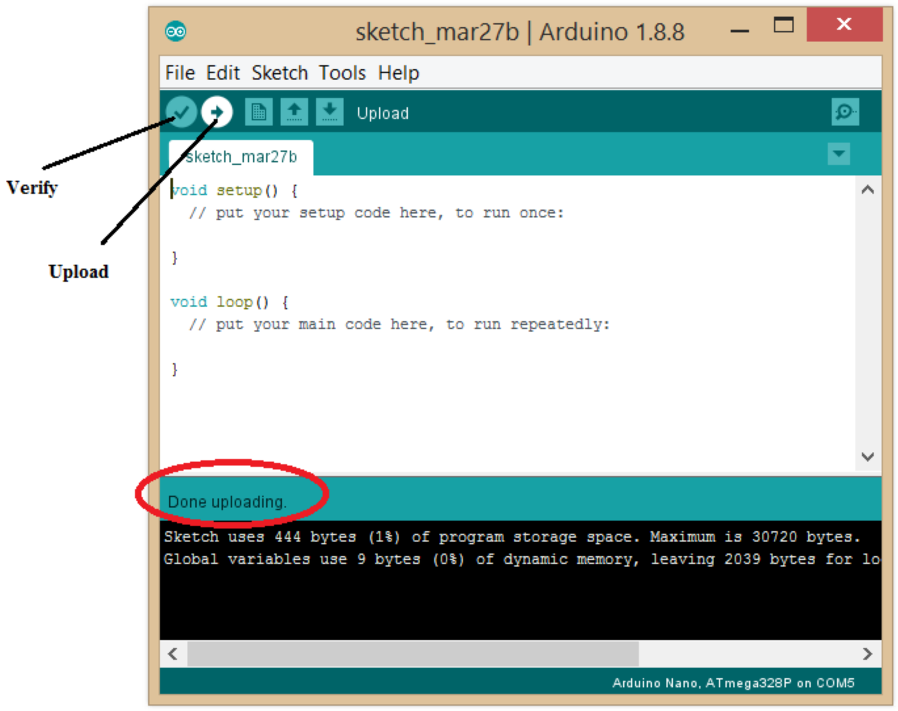
Table 1: Circuit and operation parameters for a buck converter.

4.5 Uploading code to Arduino

Once you have made the connections for Arduino and laptop. On the Arduino IDE, click on **Verify** to check if there are syntax errors in your code. Once verified, click on **Upload** to upload the code to the Arduino chip. Once it is finished, a "Done uploading" will appear, as shown in Fig. 7 to indicate the code is successfully written on the chip.

5 Equipment and Components

- 1 $\times$ Assembled testing board (Synchronous Buck Converter)
- 1 $\times$ Breadboard
- 2 $\times$ Digital multimeter (benchtop and/or hand-held)
- 1 $\times$ Laptop computer with Arduino IDE installed or Web Editor Account created (each group to prepare)


sketch_mar27b | Arduino 1.8.8
File Edit Sketch Tools Help
Upload
sketch_mar27b
Verify
Upload
Done uploading.
Arduino Nano, ATmega328P on COM5

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
Sketch uses 444 bytes (1%) of program storage space. Maximum is 30720 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for lo
```

Figure 7: Code is successfully downloaded to the Arduino chip.

- 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
- 1 × 1.8 kΩ for R_{in1}
- 1 × 8.2 kΩ for R_{in2}
- 1 × 1 kΩ for R_1
- 1 × 10 nF for C_1

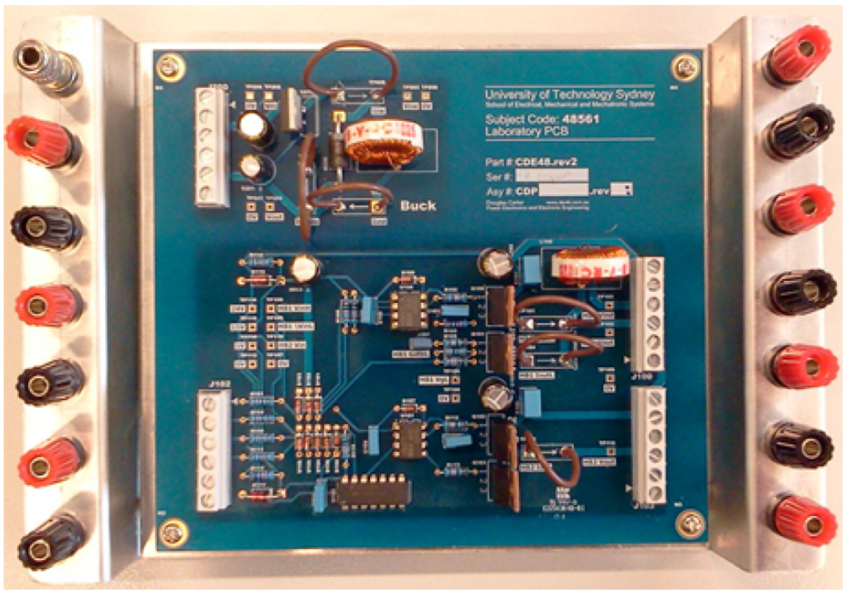


Figure 8: Assembted testing board.

6 Experiments

6.1 PV emulator

In the first experiment, you are going to plot the I-V and P-V curves of the PV emulator.

a) Set first ouput of the power supply to 0 V with the current limit at 0.4A.

b) Turn off the power supply.

c) 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.

d) Connect the power supply to the PV emulator board.

e) Turn on the power supply. Turn the voltage knob to 21V while keeping the current limit to 0.4A. Note that this time the current limit has to be more precisely tuned as it will impact the PV emulator performance.

f) Turn off the power supply.

g) Connect the resistor box to another terminals of the PV emulator board via a digital meter for current measurement.

h) Connect another digital meter for voltage measurement of the diode string.

i) Turn on the power supply. Vary the resistance values from 0 Ω to 270 Ω and also in open circuit (OC) condition. Record the current and voltage readings for each resistance value.

j) 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.

k) Repeat steps g to j using other current-limit values such as 0.3A and 0.2A while not changing the voltage knob.

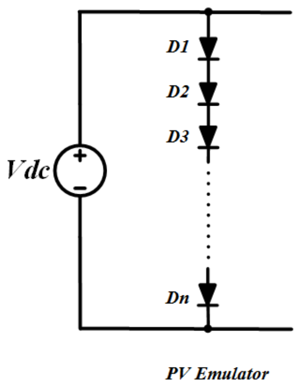


Figure 9: Circuit diagram of a PV emulator.

l) Once finished, 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.

6.2 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 in Section 4.

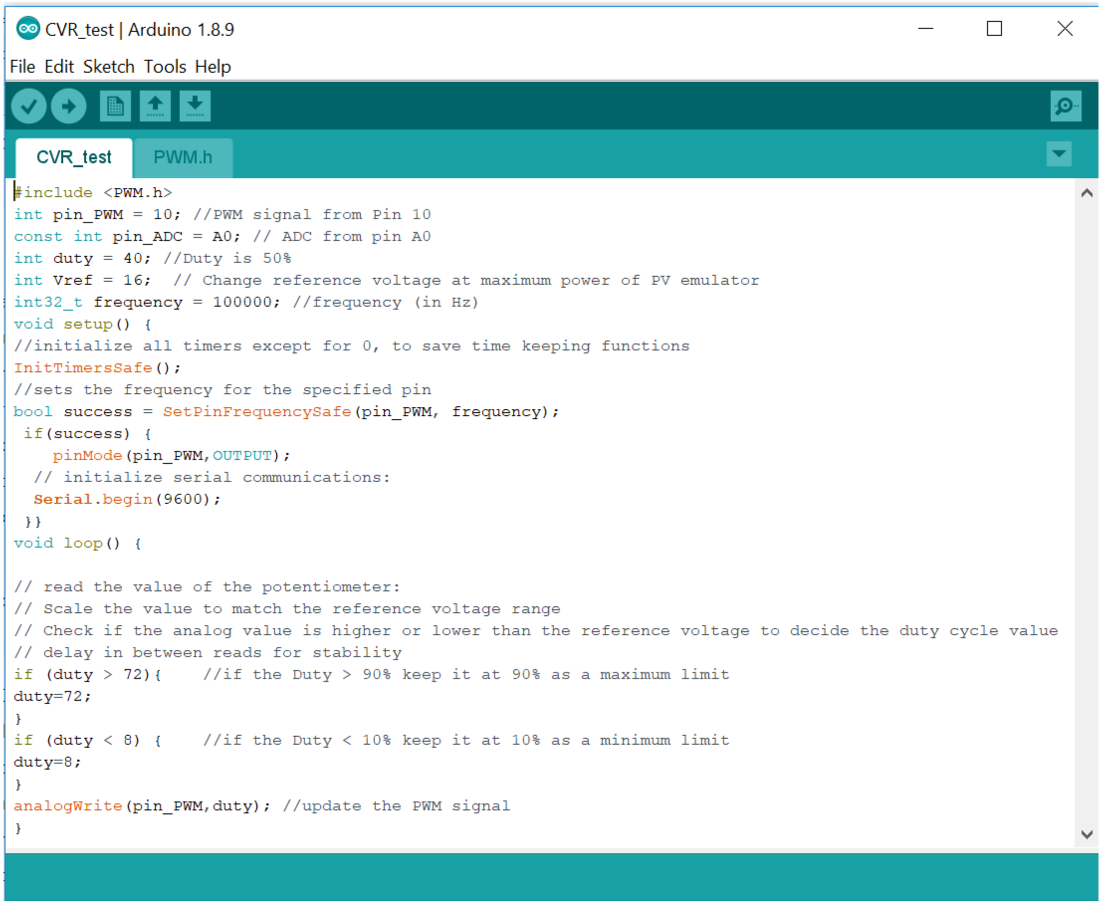
h) Observe and record the PWM signal from the PWM pin (D10) of the Arduino board using the DSO.

6.3 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.



```
#include <PWM.h>
int pin_PWM = 10; //PWM signal from Pin 10
const int pin_ADC = A0; // ADC from pin A0
int duty = 40; //Duty is 50%
int Vref = 16;  // Change reference voltage at maximum power of PV emulator
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 potentiometer:
// Scale the value to match the reference voltage range
// Check if the analog value is higher or lower than the reference voltage to decide the duty cycle value
// 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
}
```

Figure 10: Main framework of program for the Constant Voltage Reference MPPT.

d) Connect the 100µF capacitor and the resistor bank (select 33Ω) 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.

6.4 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.

6.5 Programming Constant Voltage Reference MPPT

a) Input the following code to your Arduino programming editor as shown in Fig. 10.

b) Based on part ii) of your prelab work. Enter the missing part of the code to complete the constant voltage reference MPPT.

6.6 Testing of Constant Voltage Reference MPPT

a) Make sure power supply is turned off.

b) Download the code to the Arduino.

c) Reconnect PV emulator between the 21V power supplay and the buck converter as shown in Fig. 2 and turn on the power supply.

d) Observe on the DSO the PWM signal, input current to the buck converter (i.e. clamp the lead to 21V) and input voltage to the buck converter.

e) Use the MATH function to obtain the input power (i.e. product of inductor voltage and input voltage). Record all waveforms.

f) Now vary the current limit on the 21V power supply between 0.4A and 0.6A (get a few sets e.g. 0.45A, 0.5A, 0.55A, etc.). Don't let supply voltage to go below 18V as the buck converter will stop working. Observe the input voltage and see if it changes or stays relatively constant. If latter, your constant voltage reference MPPT is working.

g) Record the input voltage, input current and duty cycle of the PWM for different current limit values.

6.7 Analysis

i) Explain, with aid of equations and diagram, how the buck converter serves as a variable resistor to the PV panel.

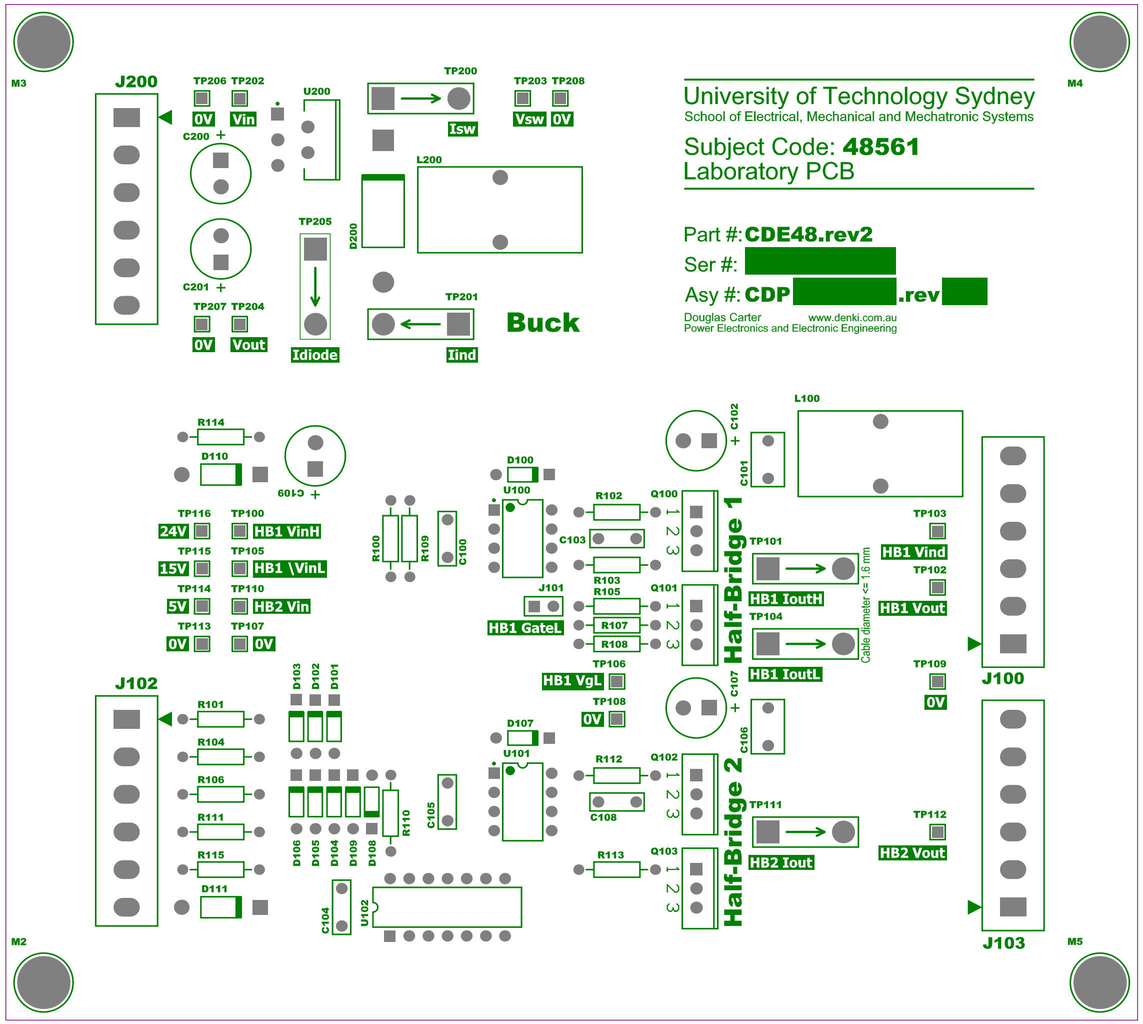
ii) Explain your part of the code to achieve constant voltage reference MPPT with the support of experimental results.

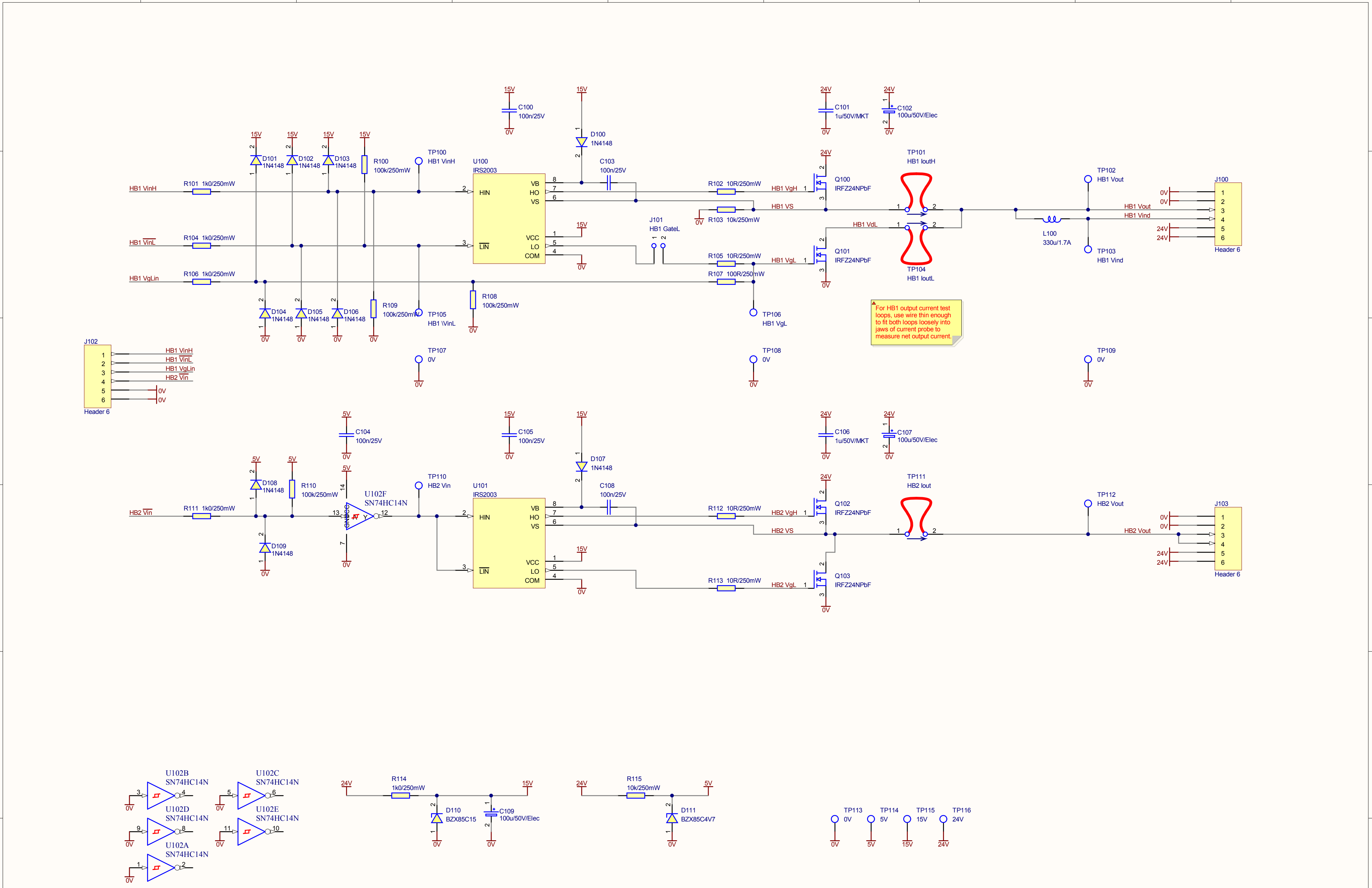
iii) Compare the constant voltage reference MPPT performance in Section 6.6 against the identified true MPPs in Section 6.1.

Appendix

(1) Top view of the Assembled Testing Board PCB

(2) Schematic diagram of the Assembled testing Board


University of Technology Sydney
School of Electrical, Mechanical and Mechatronic Systems
Subject Code: 48561
Laboratory PCB
Part #: CDE48.rev2
Ser #:
Asy #: CDP .rev
Douglas Carter www.denki.com.au
Power Electronics and Electronic Engineering
M3
M4
M2
M5
J200
TP206
TP202
0V
Vin
U200
C200
C201
TP200
Isw
TP203
TP208
Vsw
0V
L200
D200
TP205
TP207
TP204
0V
Vout
Idiode
TP201
Iind
Buck
R114
D110
C109
D100
U100
R100
R109
C100
C103
R102
Q100
R103
R105
R107
R108
Q101
J101
HB1 GateL
C102
C101
L100
TP101
HB1 IoutH
TP104
HB1 IoutL
Cable diameter <= 1.6 mm
TP103
HB1 Vind
TP102
HB1 Vout
TP109
0V
J100
Half-Bridge 1
TP116
24V
TP100
HB1 VinH
TP115
15V
TP105
HB1 \VinL
TP114
5V
TP110
HB2 Vin
TP113
0V
TP107
0V
TP106
HB1 VgL
TP108
0V
J102
R101
R104
R106
R111
R115
D111
D103
D102
D101
D106
D105
D104
D109
D108
R110
C105
C104
U102
D107
U101
C107
C106
R112
Q102
C108
R113
Q103
TP111
HB2 Iout
TP112
HB2 Vout
Half-Bridge 2
J103
1 2 3



All resistors 1%/250mW unless noted otherwise.
All capacitors 10%/50V unless noted otherwise.

Title: Power Electronics Experimental Board University of Sydney, Technology		
Document No.: DE45	Date: 14 Apr 2011	Drawn By: DRHC
Rev: 2 · Size: A3 · Sheet No.: 1 / 3	Checked By: DGD	Approved By: DRHC
File: G:\ArtShokai\Projects\PR15 UTS, Labs\Design1 Capture\Electronic\DE45 PE experiment\BridgeConverters.SchDoc		

Douglas Carter
BA PhD Camb, MIEEE
Power Electronics and Electronic Engineering
1/17 Albert Street Edgecliff NSW 2027 Australia
P +61 413 010 838 E douglas.carter@denki.com.au