Cégep de Sherbrooke

Technologie de systèmes ordinés

Documentation projet Drone

Pour le cours de ISO

Pour : M. Pierre Bergeron

Par : François Archambault

Session Automne 2020

Table des matières:

INTRODUCTION AU PROJET :

Le but de ce projet était de concevoir l'électronique d'un drone et de le faire fonctionner. Pour ce rapport, je ne vais que traiter du côté matériel du projet parce que je n'ai pas vraiment commencé la programmation et l'application en Java n'est pas commencée.

Voici le schéma fonctionnel de mon projet :

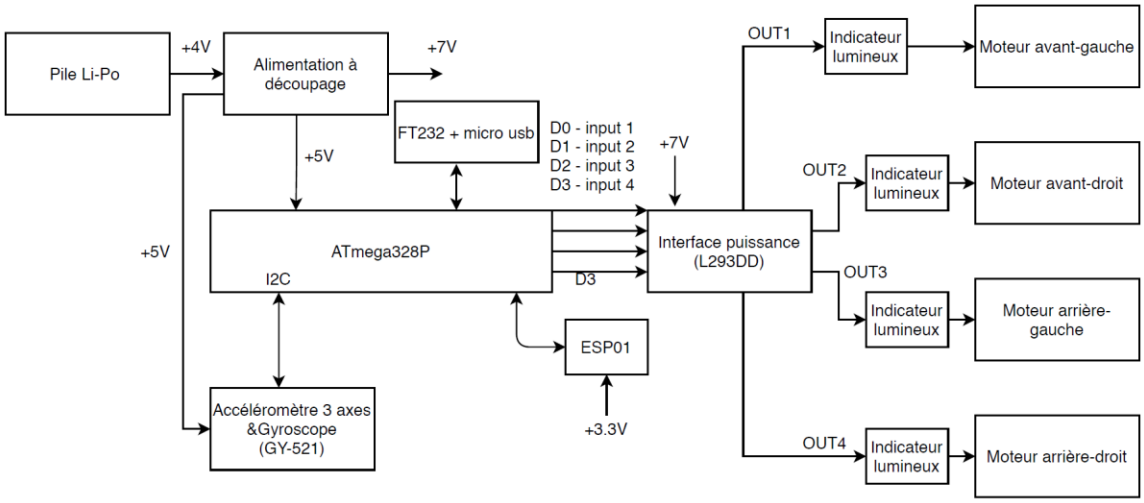


Figure 1

Il y a quelques erreurs dans ce schéma, mais malgré cela, le fonctionnement de base est comprenable. L'électronique du drone sera alimenté par une petite batterie Li-Po 4.5volts et 1.5 A/heure. Il y a un microcontrôleur qui gère le traitement des données du gyroscope, de l'esp01 et il contrôle les ponts en H des moteurs. Le gyroscope fournit au microcontrôleur des données sur l'accélération et la position en x et y du drone. L'esp01 gère la connexion Wi-Fi avec la future application Java sur cellulaire. Plusieurs témoins lumineux sont présents pour indiquer dans quel sens les moteurs tournent. L'alimentation à découpage monte la tension à 7v ou 9v pour alimenter les moteurs (en fait les ponts en H).

CHOIX DES TECHNOLOGIES :

J'ai décidé de travailler avec un Atmega328PB parce que je commence à bien le connaitre et souvent, dans d'autres projets, j'ai dû travailler avec ce microcontrôleur. L'Atmega328PB a assez de pins PWM pour permettre le bon fonctionnement du drone. Pour le projet, j'ai mis le bootloader de Arduino pour pouvoir utiliser l'IDE et les librairies d'Arduino.

Les autres pièces ont été suggérées par le professeur. Honnêtement, je n'aime pas beaucoup l'esp01 parce qu'il n'est vraiment pas stable et il y a souvent des problèmes de communications. Mais bon… Ça marche quand même et ça ne coûte presque rien!

Je ne connaissais pas le gyroscope, mais j'avais déjà utilisé les ponts en H dans un projet personnel (bateau téléguidé).

Explications du schéma électrique :

(Le schéma électrique se trouve dans l'annexe 2 à la fin du document)

Le schéma électrique a été fait avec KiCad et il y a 4 parties au schéma. Les parties sont : alimentation, microcontrôleur et gyroscope, moteurs et esp01.

Alimentation : Dans cette partie, toutes les connexions de l'alimentation à découpage sont présentes ainsi que les régulateurs 5v et 3v3. Il y a aussi le jumper qui permet de choisir entre une tension de sortie de 7 ou 9 volts. (Figure 2)

Microcontrôleur : Dans cette partie, il y a les connexions du FT232 qui sert à la communication série entre un ordinateur et l'Atmega328PB. Il y a aussi le gyroscope qui est connecté au microcontrôleur en I2C. Il y a aussi les jumpers qui servent à contrôler les liens séries entre l'esp01, le FT232 et l'Atmega328PB. (Figure 3)

Moteurs : Dans cette partie, il y a les moteurs, les témoins lumineux et les ponts en H. Plusieurs pins des ponts en H sont connectés à des sorties du microcontrôleur (Les Dx ou Ax). Pour réussir à appliquer des PWMs sur les moteurs, il faut utiliser les pins enables des ponts en H et les pins inputs. Les pins inputs servent à choisir sur quel moteur le PWM va être appliqué et c'est sur les pins enables que le PWM va être transmis au moteur. Exemple : Le moteur 1 est connecté à l'input 1 et 2 d'un pont en H. Si l'input #1 est à un niveau haut, le moteur va aller dans le sens antihoraire et si c'est l'input #2 qui est à un niveau haut, le moteur va aller dans le sens horaire. Cependant, pour fonctionner il faut que le pin enable du moteur soit à un niveau haut! Si un PWM est appliqué sur la pin enable, le PWM va se retrouver dans le moteur. (Figure 4)

Esp01 : Dans cette partie, il n'y a que l'esp01 avec les boutons pour le programmer et une del témoin pour démontrer que la connexion est bien présente. (Figure 5)

Problèmes rencontrés :

Plusieurs problèmes sur le matériel sont survenus durant la conception. Dans la première version du drone, il y avait plusieurs erreurs de branchements entre le microcontrôleur et les ponts en H qui contrôlent les moteurs. Dans la première version, l'ATmega328PB n'aurait pas pu générer des PWMs sur tous les sens des moteurs. Malgré les erreurs, le PCB a été fabriqué et assemblé. Le PCB était fonctionnel sauf pour les PWMs. Le microcontrôleur fonctionnait bien ainsi que la communication série. Le PCB a malheureusement surchauffé à cause d'une erreur de son concepteur en branchant la batterie. Le cavalier qui donnait une référence (7v ou 9v) pour le circuit d'alimentation du drone a été enlevé et le circuit a brulé.

Il a donc fallu une version 2 du PCB avec les corrections nécessaires pour le fonctionnement des PWMs. Le PCB fut monté et à la plus grande joie du concepteur, tout est fonctionnel dans la version 2 de la plaque.

ESP01 :

Dans le projet du drone, l'esp01 sert à faire le lien entre l'application cellulaire qui va envoyer les changements voulus et le microcontrôleur qui contrôle les moteurs du drone. L'esp01 reçoit des trames par le port série. Premièrement, l'utilisateur doit connecter l'application avec l'esp01 du drone. Il doit aller dans la section configuration de l'application et entrer l'IP, le port et l'ID pour la connexion (Dans ce projet : Port : 8888 & ID : crap8266). Si l'utilisateur demande de changer la direction du drone en appuyant sur une direction (4 boutons sur l'application), la trame envoyée à l'esp01 par l'application sera la direction avec une petite étoile (ex : Front*). Si l'application envoie une vitesse (avec le curseur), la vitesse sera envoyée avec un « / » (ex : 56/).

Quand la trame vient du microcontrôleur, les informations ne sont plus des directions ou bien une vitesse, mais les valeurs X, Y, Z de l'accéléromètre qui se trouve sur la plaque du drone. La trame reçue ressemble à ça : « 1234 5667 6374/*». La première section de la trame est le X, la deuxième Y et finalement la troisième Z. La trame se termine avec « /* » parce que dans le code de l'esp01, l'esp01 écoute le port série jusqu'à temps qu'il voit une petite étoile (*). Une fois qu'il voit la petite étoile, il transmet la trame à l'application CodeNameOne. La trame envoyée sera donc : « 1234 5667 6374/».

ATmega328PB :

Dans le projet, l'Atmega328PB est le microcontrôleur qui contrôle les moteurs (donc PWM) du drone. Il active les moteurs avec les trames qu'il reçoit de l'application CodeNameOne avec le lien que fait l'esp01. Les trames reçues ressemblent à cela : « Front*» ou bien «76/». Le programme de l'ATmega328PB écoute le port série qui est connecté avec l'esp01. Quand quelque chose arrive par le port série, les caractères sont mis un à un dans une variable string (command). Si le caractère reçu est une petite étoile (*), cela veut dire que c'est une direction qui a été envoyée et l'ATmega328PB fait les ajustements par rapport à la commande. S'il reçoit la commande « Front », il va faire avancer le drone.

Si le caractère reçu est « / », cela veut dire que c'est une vitesse qui a été envoyée. La vitesse influence le PWM généré sur les moteurs.

Quand il a reçu une trame, l'ATmega328PB construit une variable string (valeurACC) qui contient les valeurs X, Y et Z de l'accéléromètre. Cette variable va être envoyée à l'esp01 par le port série pour finalement être reçu par l'application sur cellulaire. L'application pourra ensuite afficher les données de l'accéléromètre.

Application CodeNameOne :

L'application cellulaire fait avec CodeNameOne et l'IDE NetBeans, sert à contrôler le drone à partir d'un cellulaire. Dans la fenêtre principale de l'application (Annexe 4 pour l'apparence de l'application), il y a trois parties. La première (en haut) permet de contrôler la direction du drone grâce aux quatre boutons qui figurent les directions Nord, Sud, Est et Ouest. Quand l'utilisateur

appuie sur un des quatre boutons, l'application envoie une requête TCP vers l'esp01 qui se trouve sur le drone. Si l'utilisateur appuie sur le bouton qui fait avancer le drone, la commande (ou trame) envoyée sera : « Front* ».

La deuxième partie (curseur) sert à changer la vitesse du drone. Le curseur peut aller de 0 à 100. La trame envoyée sera : « 21/ ».

La troisième partie sert à afficher les valeurs de l'accéléromètre du drone (X, Y, Z).

Cependant, avant de pouvoir envoyer des commandes au drone, il faut connecter l'application avec le esp01. Pour cela, il faut aller dans le menu (3 petits points en haut à droite) et choisir l'onglet Configuration. Dans cette fenêtre, il est possible d'entrer l'IP, le port et l'ID de l'esp01 du drone. Si une des trois informations n'est pas valides, il n'y aura pas de connexion. L'ID sert à vérifier si les commandes arrivent bien du drone sur lequel l'application est connectée.

Sinon, dans le menu, il y a aussi un onglet « À propos de… » et un onglet « Quitter ». En cliquant sur l'onglet « À propos de… », un message apparait pour montrer la version de l'application et le nom du concepteur. En cliquant sur l'onglet « Quitter », l'application se ferme.

TCP :

Le drone et l'application communiquent avec l'aide de trames. Ces trames sont envoyées avec le protocole de communication TCP. L'esp01 est en réalité un serveur TCP qui attend qu'un client se connecte. L'application joue le rôle du client. À chaque fois qu'une trame est envoyée à partir de l'application, l'application se connecte au serveur TCP de l'esp01 et une fois que le commande a été envoyée, le client (donc l'application) se déconnecte du serveur. De son côté, l'esp01 est toujours en attente d'une connexion d'un client. Quand un client est connecté, il reçoit la trame, il envoie le contenu de la trame sur le port série et il attend une réponse du microcontrôleur. Une fois que le microcontrôleur a envoyé sa réponse, l'esp01 envoie le contenu de la réponse au client (l'application).

FONCTIONNALITÉS FINALES OBTENUES :

Le drone fonctionne bien. Toutes les parties ont été évaluées. Les codes nécessaires au fonctionnement du drone sont aussi fonctionnels, mais il y a quand même une petite erreur dans le programme de l'application CodeNameOne. Si l'utilisateur connecte correctement l'application à l'esp01 et qu'ensuite, il décide de changer l'ID sans changer le reste, le drone va continuer à recevoir les commandes malgré le mauvais ID. Cependant, si l'utilisateur tente de mettre un mauvais ID dans le première connexion à l'esp01, l'accès sera refusé.

ANNEXE 1 (SIMULATION 7 ET 9V) :

Simulation 7v :

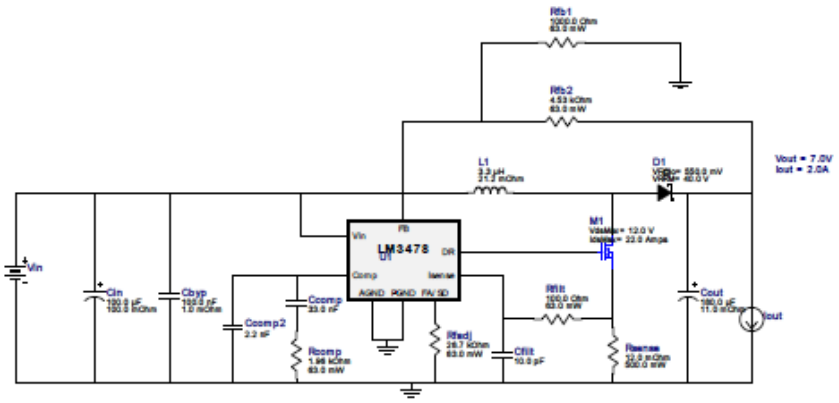
TEXAS INSTRUMENTS

WEBENCH® Design Report

VinMin = 3.5V
VinMax = 4.2V
Vout = 7.0V
Iout = 2.0A

Device = LM3478MMX/NOPB
Topology = Boost
Created = 2020-03-31 18:49:58.351
BOM Cost = $2.78
BOM Count = 16
Total Pd = 1.79W

Design : 3 LM3478MMX/NOPB
LM3478MMX/NOPB 3.5V-4.2V to 7.00V @ 2A

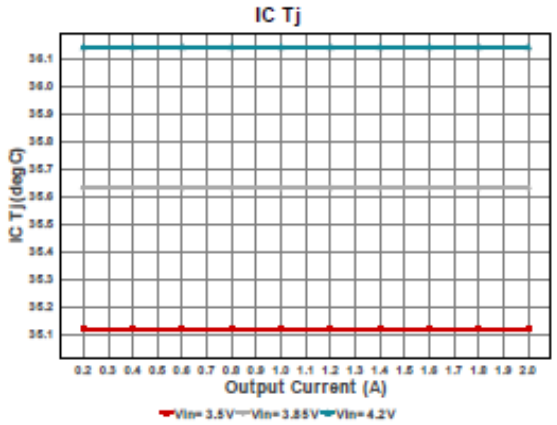


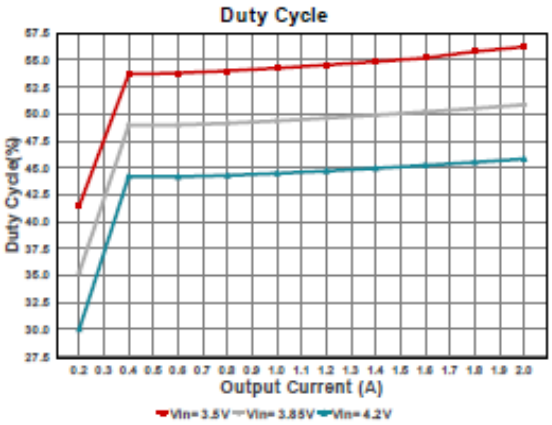
1. With the low turn of voltage of the LM34x8 your power supply may current limit before you reach your working input voltage. If this happens, or to preempt this from happening, you can include a low pass RC filter from input voltage to Vin on the IC. Make sure the rise time on the RC network is slower than your supply's rise time.

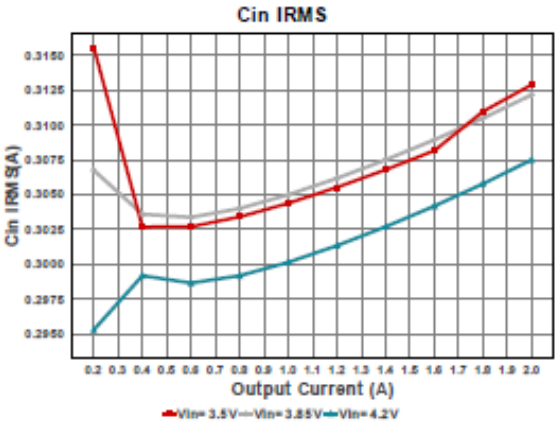
Electrical BOM

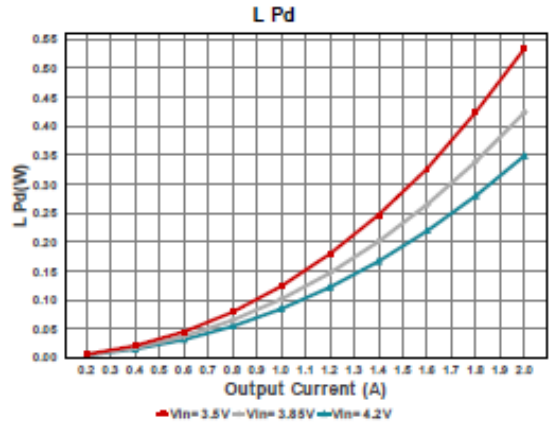
Name	Manufacturer	Part Number	Properties	Qty	Price	Footprint
Cbyp	MuRata	GRM155R70J104KA01D Series= X7R	Cap= 100.0 nF ESR= 1.0 mOhm VDC= 6.3 V IRMS= 0.0 A	1	$0.01	0402 3 mm²
Ccomp	TDK	CGA4J2C0G1H333J125AA Series= C0G/NP0	Cap= 33.0 nF VDC= 50.0 V IRMS= 0.0 A	1	$0.09	0805 7 mm²
Ccomp2	Samsung Electro-Mechanics	CL21C222JBFNNNE Series= C0G/NP0	Cap= 2.2 nF VDC= 50.0 V IRMS= 0.0 A	1	$0.03	0805 7 mm²
Cfilt	Samsung Electro-Mechanics	CL21C100JBANNNC Series= C0G/NP0	Cap= 10.0 pF VDC= 50.0 V IRMS= 0.0 A	1	$0.01	0805 7 mm²
Cin	Panasonic	6TPH100MAEA Series= TPH	Cap= 100.0 uF ESR= 100.0 mOhm VDC= 6.3 V IRMS= 670.0 mA	1	$0.32	CAPSMT_6_A09 11 mm²
Cout	Panasonic	16SVPE180M Series= SVPE	Cap= 180.0 uF ESR= 11.0 mOhm VDC= 16.0 V IRMS= 4.46 A	1	$0.50	CAPSMT_62_C10 74 mm²
D1	Diodes Inc.	B540C-13-F	VF@Io= 550.0 mV VRRM= 40.0 V	1	$0.17	SMC 83 mm²
L1	Coilcraft	XAL5030-332MEB	L= 3.3 µH 21.2 mOhm	1	$0.63	XAL5030 54 mm²

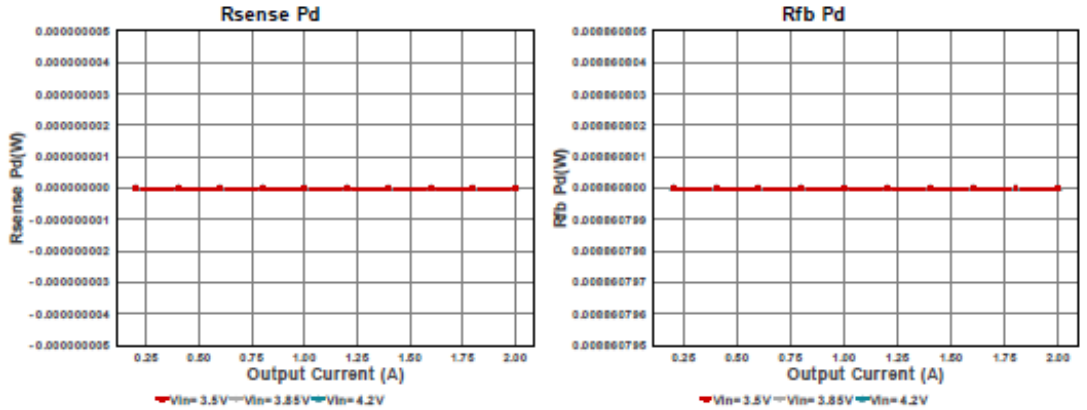
Name	Manufacturer	Part Number	Properties	Qty	Price	Footprint
M1	Texas Instruments	CSD13202Q2	VdsMax= 12.0 V IdsMax= 22.0 Amps	1	$0.12	DQK0006C 9 mm²
Rcomp	Vishay-Dale	CRCW04021K96FKED Series= CRCW..e3	Res= 1.96 kOhm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	0402 3 mm²
Rfadj	Vishay-Dale	CRCW040226K7FKED Series= CRCW..e3	Res= 26.7 kOhm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	0402 3 mm²
Rfb1	Vishay-Dale	CRCW04021K00FKED Series= CRCW..e3	Res= 1000.0 Ohm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	0402 3 mm²
Rfb2	Vishay-Dale	CRCW04024K53FKED Series= CRCW..e3	Res= 4.53 kOhm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	0402 3 mm²
Rfilt	Vishay-Dale	CRCW0402100RFKED Series= CRCW..e3	Res= 100.0 Ohm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	0402 3 mm²
Rsense	Stackpole Electronics Inc	CSR1206FK12L0 Series= ?	Res= 12.0 mOhm Power= 500.0 mW Tolerance= 1.0%	1	$0.12	1206 11 mm²
U1	Texas Instruments	LM3478MMX/NOPB	Switcher	1	$0.73	MUA08A 24 mm²

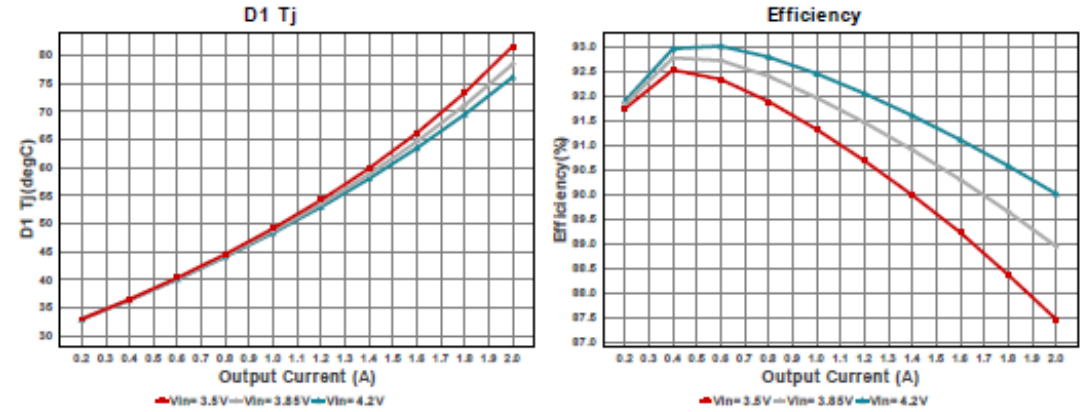


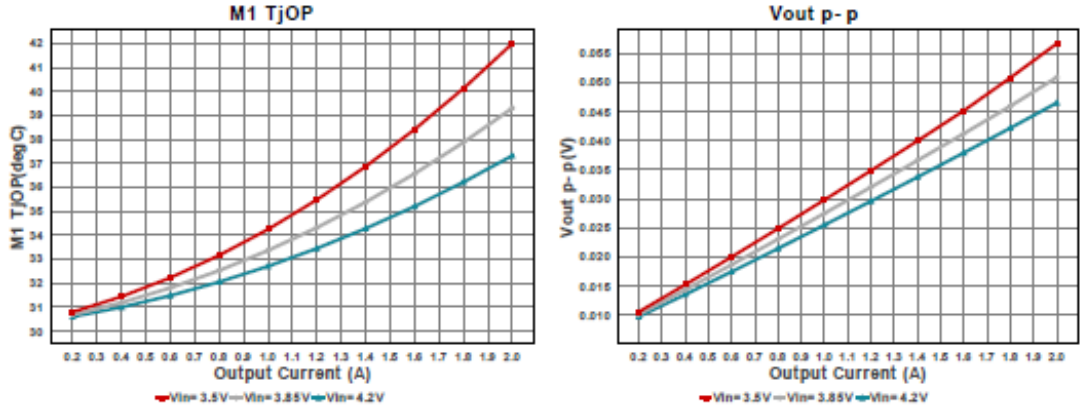


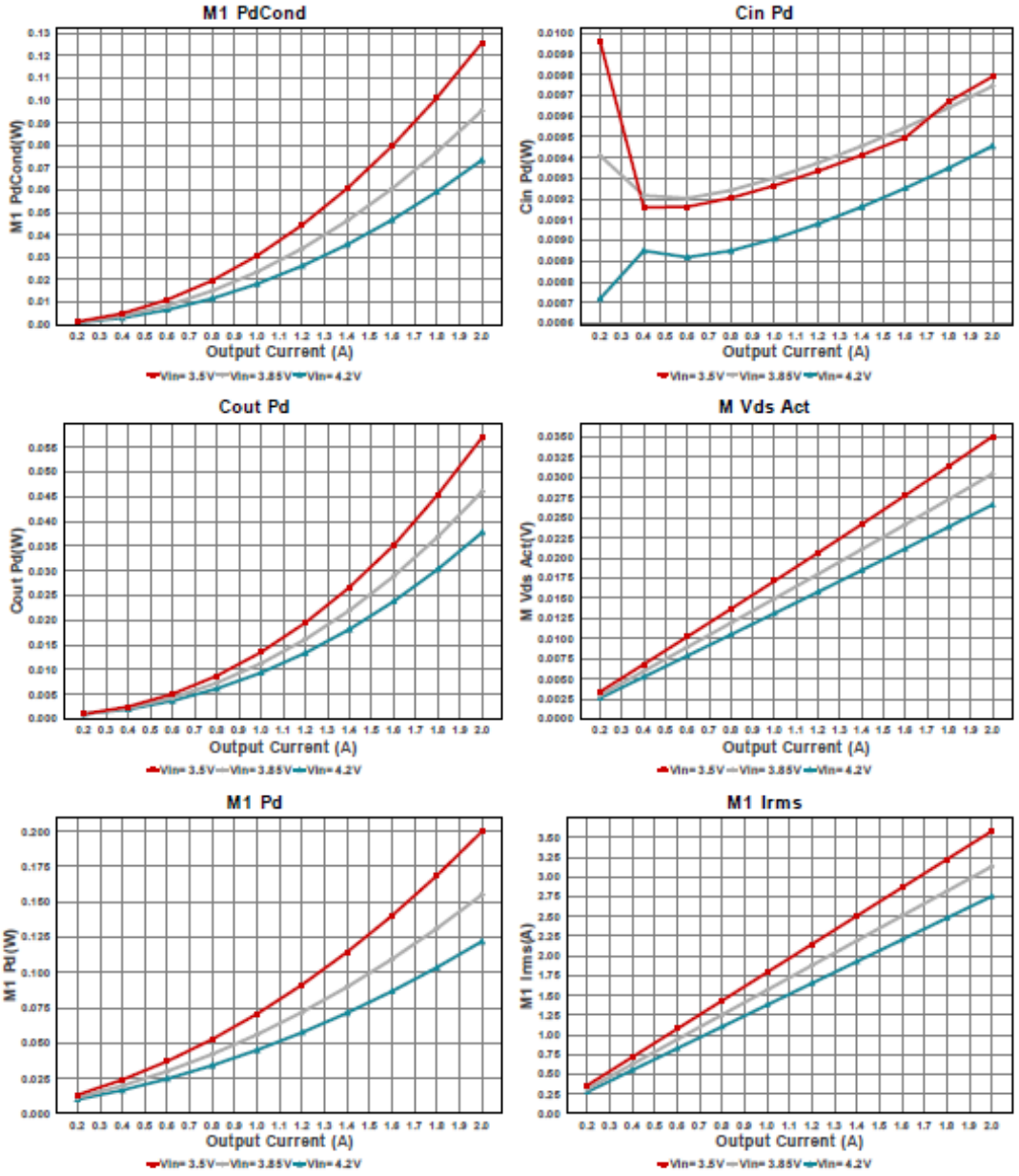


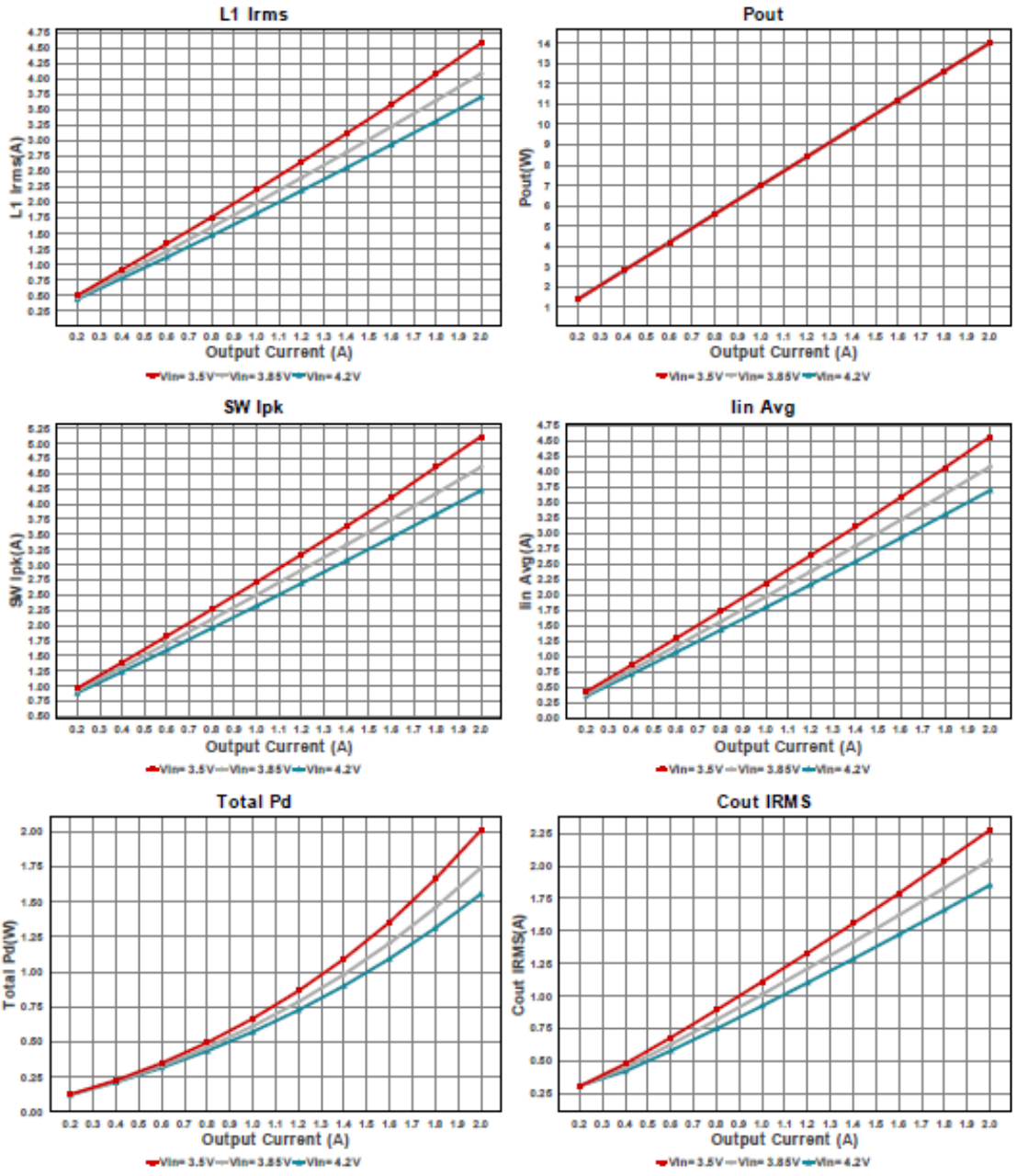


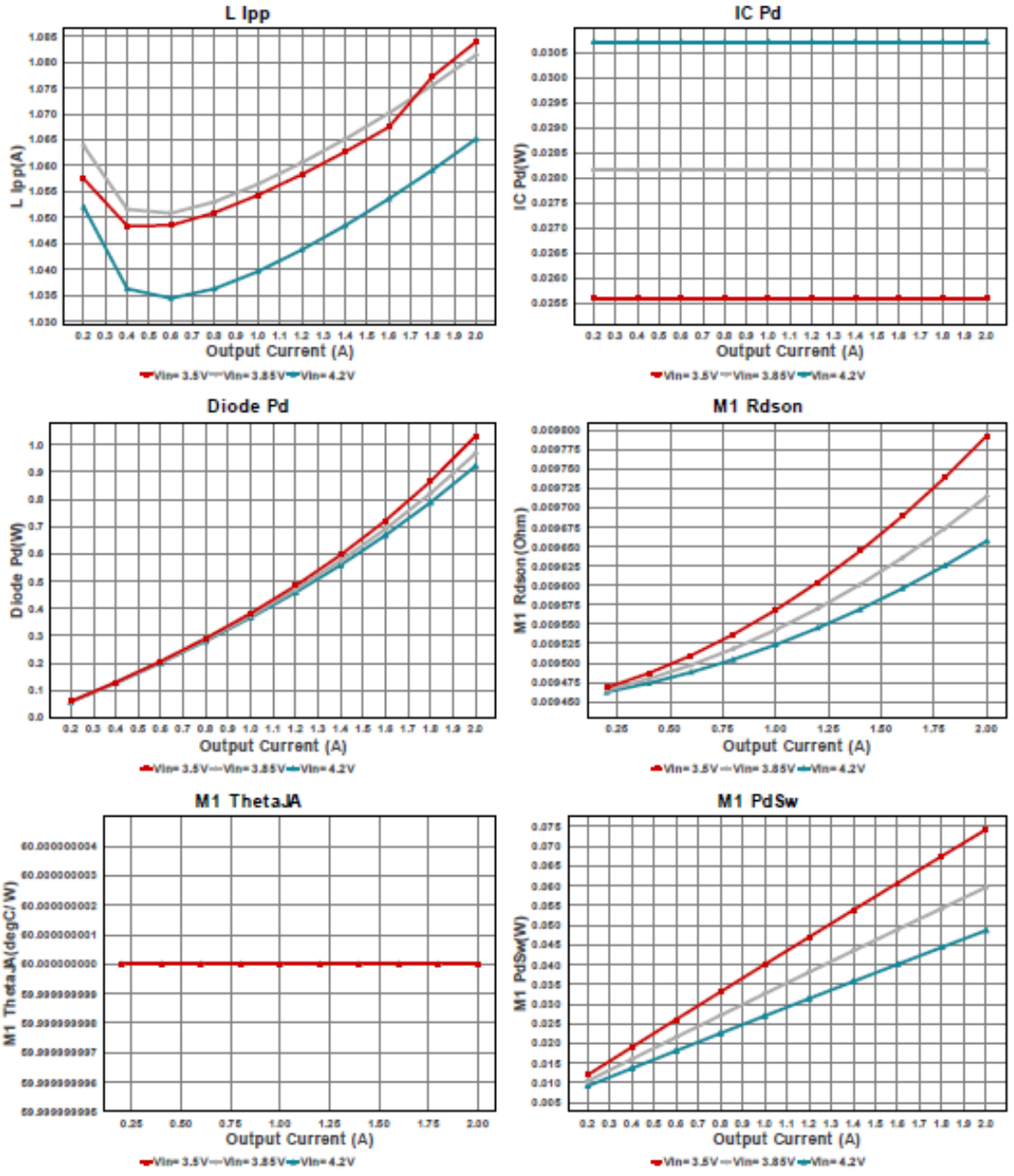


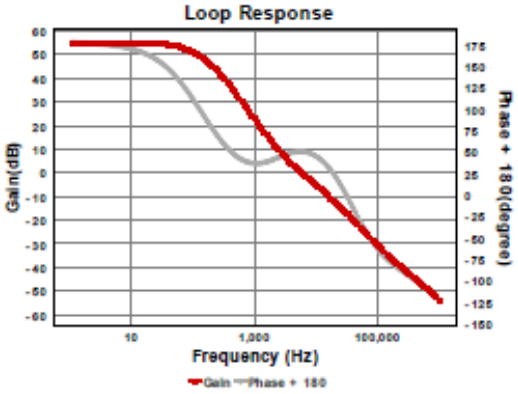












Operating Values

#	Name	Value	Category	Description
1.	Cin IRMS	309.574 mA	Capacitor	Input capacitor RMS ripple current
2.	Cin Pd	9.584 mW	Capacitor	Input capacitor power dissipation
3.	Cout IRMS	2.249 A	Capacitor	Output capacitor RMS ripple current
4.	Cout Pd	55.646 mW	Capacitor	Output capacitor power dissipation
5.	D1 Tj	81.6 degC	Diode	D1 junction temperature
6.	Diode Pd	1.032 W	Diode	Diode power dissipation
7.	IC Pd	25.781 mW	IC	IC power dissipation
8.	IC Tj	35.156 degC	IC	IC junction temperature
9.	IC Tolerance	24.3 mV	IC	IC Feedback Tolerance
10.	ICThetaJA	200.0 degC/W	IC	IC junction-to-ambient thermal resistance
11.	Iin Avg	4.511 A	IC	Average input current
12.	L Ipp	1.072 A	Inductor	Peak-to-peak inductor ripple current
13.	L Pd	519.46 mW	Inductor	Inductor power dissipation
14.	L1 Irms	4.519 A	Inductor	Inductor ripple current
15.	M Vds Act	35.084 mV	Mosfet	M Vds
16.	M1 Irms	3.582 A	Mosfet	M1 MOSFET Irms
17.	M1 Pd	201.05 mW	Mosfet	M1 MOSFET total power dissipation
18.	M1 PdCond	125.66 mW	Mosfet	M1 MOSFET conduction losses
19.	M1 PdSw	75.392 mW	Mosfet	M1 MOSFET switching losses
20.	M1 Rdson	9.795 mOhm	Mosfet	Drain-Source On-resistance
21.	M1 ThetaJA	60.0 degC/W	Mosfet	MOSFET junction-to-ambient thermal resistance
22.	M1 TjOP	42.063 degC	Mosfet	M1 MOSFET junction temperature
23.	Cin Pd	9.584 mW	Power	Input capacitor power dissipation
24.	Cout Pd	55.646 mW	Power	Output capacitor power dissipation
25.	Diode Pd	1.032 W	Power	Diode power dissipation
26.	IC Pd	25.781 mW	Power	IC power dissipation
27.	L Pd	519.46 mW	Power	Inductor power dissipation
28.	M1 Pd	201.05 mW	Power	M1 MOSFET total power dissipation
29.	M1 PdCond	125.66 mW	Power	M1 MOSFET conduction losses
30.	M1 PdSw	75.392 mW	Power	M1 MOSFET switching losses
31.	Rfb Pd	8.861 mW	Power	Rfb Power Dissipation
32.	Rsense Pd	221.35 mW	Power	LED Current Rsns Power Dissipation
33.	Total Pd	1.787 W	Power	Total Power Dissipation
34.	Rfb Pd	8.861 mW	Resistor	Rfb Power Dissipation
35.	Rsense Pd	221.35 mW	Resistor	LED Current Rsns Power Dissipation
36.	BOM Count	16	System Information	Total Design BOM count
37.	Cross Freq	4.955 kHz	System Information	Bode plot crossover frequency
38.	Duty Cycle	55.635 %	System Information	Duty cycle
39.	Efficiency	88.681 %	System Information	Steady state efficiency
40.	FootPrint	304.0 mm^2	System Information	Total Foot Print Area of BOM components
41.	Frequency	541.734 kHz	System Information	Switching frequency
42.	Gain Marg	-15.966 dB	System Information	Bode Plot Gain Margin
43.	Iout	2.0 A	System Information	Iout operating point

#	Name	Value	Category	Description
44.	Low Freq Gain	53.364 dB	System Information	Gain at 1Hz
45.	Mode	CCM	System Information	Conduction Mode
46.	Phase Marg	49.462 deg	System Information	Bode Plot Phase Margin
47.	Pout	14.0 W	System Information	Total output power
48.	SW Ipk	5.044 A	System Information	Peak switch current
49.	Total BOM	$2.78	System Information	Total BOM Cost
50.	Vin	3.5 V	System Information	Vin operating point
51.	Vout	7.0 V	System Information	Operational Output Voltage
52.	Vout Actual	6.968 V	System Information	Vout Actual calculated based on selected voltage divider resistors
53.	Vout Tolerance	3.615 %	System Information	Vout Tolerance based on IC Tolerance (no load) and voltage divider resistors if applicable
54.	Vout p-p	55.92 mV	System Information	Peak-to-peak output ripple voltage

Design Inputs

Name	Value	Description
Iout	2.0	Maximum Output Current
VinMax	4.2	Maximum Input voltage
VinMin	3.5	Minimum Input voltage
Vout	7.0	Output Voltage
base_pn	LM3478	Base Product Number
source	DC	Input Source Type
Ta	30.0	Ambient temperature

WEBENCH® Assembly

Component Testing

Some published data on components in datasheets such as Capacitor ESR and Inductor DC resistance is based on conservative values that will guarantee that the components always exceed the specification. For design purposes it is usually better to work with typical values. Since this data is not always available it is a good practice to measure the Capacitance and ESR values of Cin and Cout, and the Inductance and DC resistance of L1 before assembly of the board. Any large discrepancies in values should be electrically simulated in WEBENCH to check for instabilities and thermally simulated in WebTHERM to make sure critical temperatures are not exceeded.

Soldering Component to Board

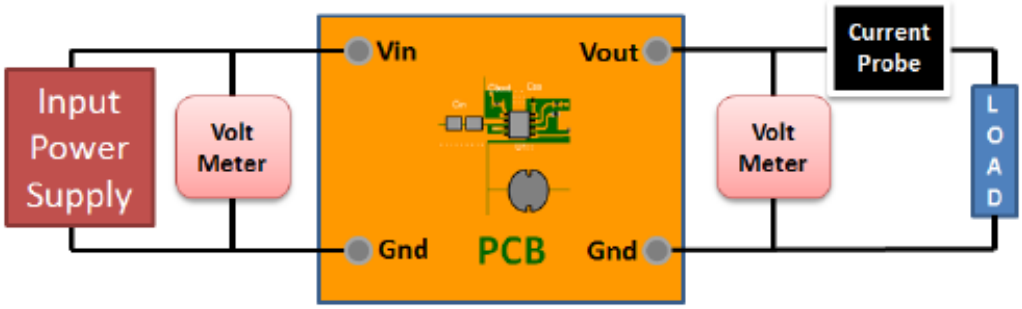
If board assembly is done in house it is best to tack down one terminal of a component on the board then solder the other terminal. For surface mount parts with large tabs, such as the DPAK, the tab on the back of the package should be pre-tinned with solder, then tacked into place by one of the pins. To solder the tab down to the board place the iron down on the board while resting against the tab, heating both surfaces simultaneously. Apply light pressure to the top of the plastic case until the solder flows around the part and the part is flush with the PCB. If the solder is not flowing around the board you may need a higher wattage iron (generally 25W to 30W is enough).

Initial Startup of Circuit

It is best to initially power up the board by setting the input supply voltage to the lowest operating input voltage 3.5V and set the input supply's current limit to zero. With the input supply off connect up the input supply to Vin and GND. Connect a digital volt meter and a load if needed to set the minimum Iout of the design from Vout and GND. Turn on the input supply and slowly turn up the current limit on the input supply. If the voltage starts to rise on the input supply continue increasing the input supply current limit while watching the output voltage. If the current increases on the input supply, but the voltage remains near zero, then there may be a short or a component misplaced on the board. Power down the board and visually inspect for solder bridges and recheck the diode and capacitor polarities. Once the power supply circuit is operational then more extensive testing may include full load testing, transient load and line tests to compare with simulation results.

Load Testing

The setup is the same as the initial startup, except that an additional digital voltmeter is connected between Vin and GND, a load is connected between Vout and GND and a current meter is connected in series between Vout and the load. The load must be able to handle at least rated output power + 50% (7.5 watts for this design). Ideally the load is supplied in the form of a variable load test unit. It can also be done in the form of suitably large power resistors. When using an oscilloscope to measure waveforms on the prototype board, the ground leads of the oscilloscope probes should be as short as possible and the area of the loop formed by the ground lead should be kept to a minimum. This will help reduce ground lead inductance and eliminate EMI noise that is not actually present in the circuit.

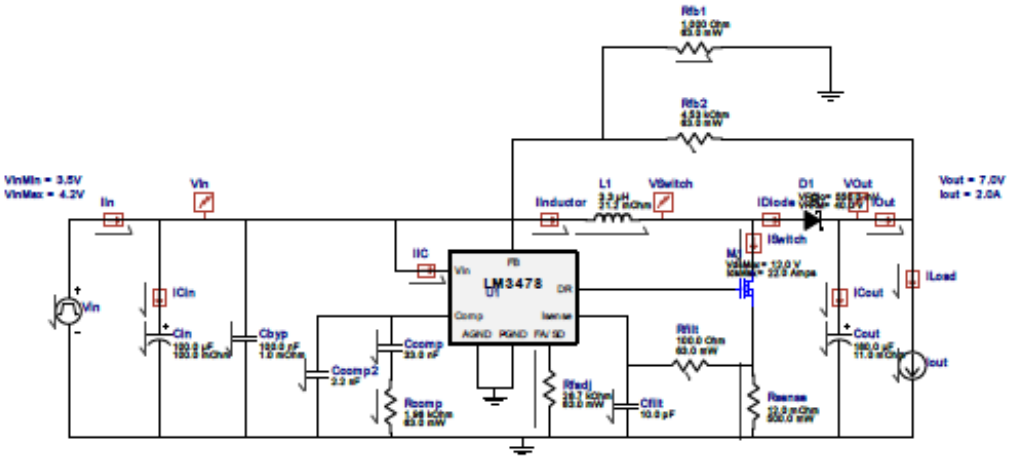


WEBENCH® Electrical Simulation Report

Design Id = 3

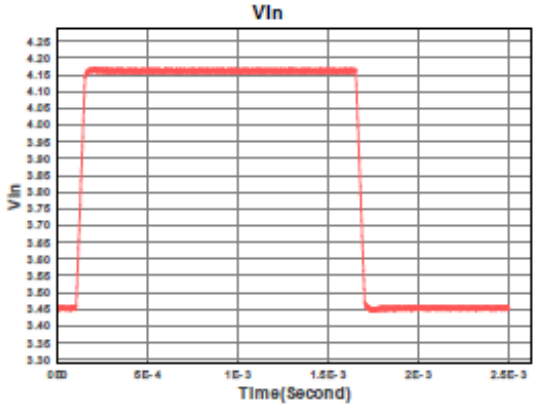
sim_Id = 1

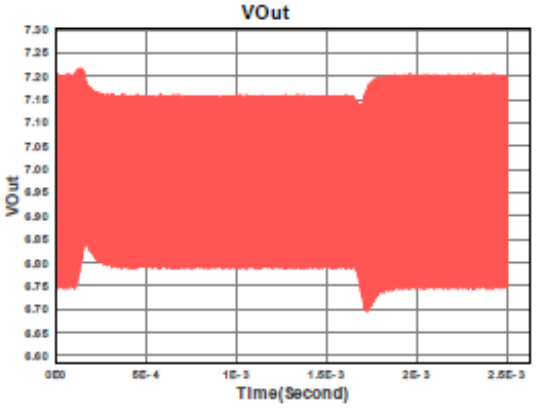
Simulation Type = Input Transient

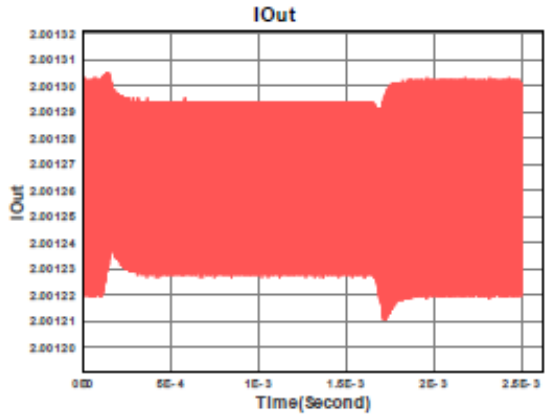


Simulation Parameters

#	Name	Parameter Name	Description	Values
1.	Iout	I	Load Current	2.0 A







Design Assistance

1. Master key : E51B955EA55D6907[v1]
2. **LM3478** Product Folder : http://www.ti.com/product/LM3478 : contains the data sheet and other resources.

Simulation 9v :

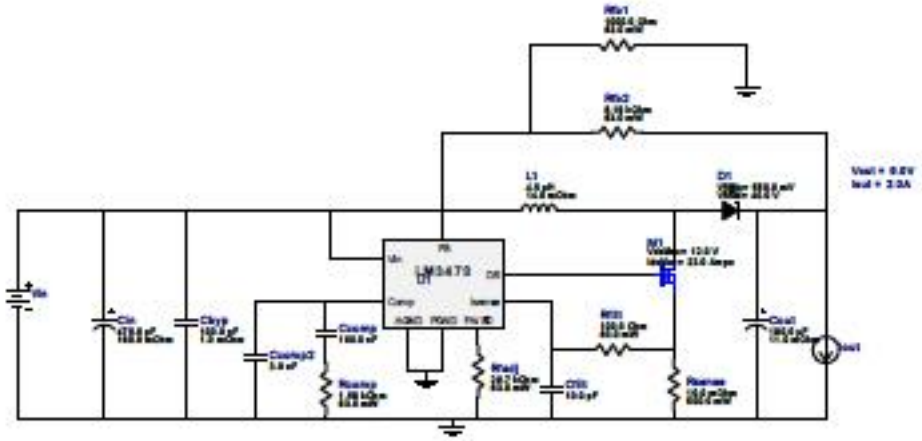
TEXAS INSTRUMENTS

WEBENCH® Design Report

Design : 4 LM3478MMDX/NOPB

VinMin = 3.5V
VinMax = 4.2V
Vout = 9.0V
Iout = 2.0A

Device = LM3478MMDX/NOPB
Topology = Boost
Created = 2020-08-28 13:56:47.870
BOM Cost = $2.40
BOM Count = 16
Total Pd = 1.98W



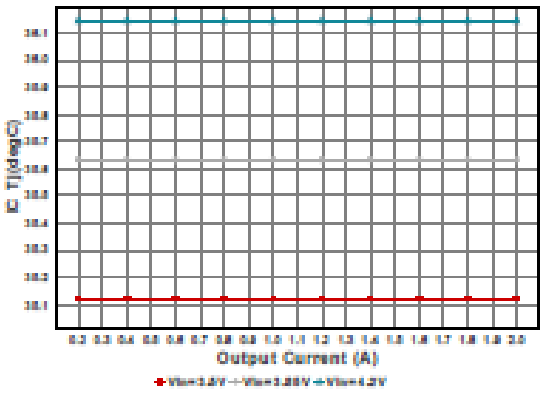
1. With the low turn of voltage of the LM34x8 your power supply may current limit before you reach your working input voltage. If this happens, or to preempt this from happening, you can include a low pass RC filter from input voltage to Vin on the IC. Make sure the rise time on the RC network is slower than your supply's rise time.

Electrical BOM

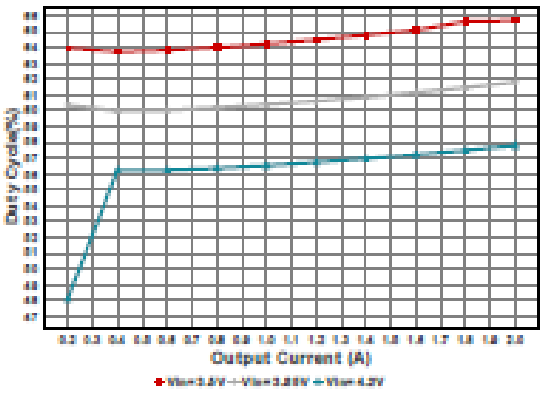
Name	Manufacturer	Part Number	Properties	Qty	Price	Footprint
Cbyp	MuRata	GRM155R70J104KA01D Series= X7R	Cap= 100.0 nF ESR= 1.0 mOhm VDC= 6.3 V IRMS= 0.0 A	1	$0.01	0402 3 mm²
Ccomp	AVX	08053C104JAZ2A Series= X7R	Cap= 100.0 nF VDC= 25.0 V IRMS= 0.0 A	1	$0.06	0805 7 mm²
Ccomp2	TDK	CGA4C2C0G1H392J060AA Series= C0G/NP0	Cap= 3.9 nF VDC= 50.0 V IRMS= 0.0 A	1	$0.04	0805 7 mm²
Cfilt	Samsung Electro-Mechanics	CL21C100JBANNNC Series= C0G/NP0	Cap= 10.0 pF VDC= 50.0 V IRMS= 0.0 A	1	$0.01	0805 7 mm²
Cin	Panasonic	EEE-FK0J471P Series= FK	Cap= 470.0 uF ESR= 160.0 mOhm VDC= 6.3 V IRMS= 500.0 mA	1	$0.14	SM_RADIAL_F 124 mm²
Cout	Panasonic	16SVPE180M Series= SVPE	Cap= 180.0 uF ESR= 11.0 mOhm VDC= 16.0 V IRMS= 4.46 A	1	$0.50	CAPSMT_62_C10 74 mm²
D1	Diodes Inc.	B540C-13-F	VF@Io= 550.0 mV VRRM= 40.0 V	1	$0.17	SMC 83 mm²

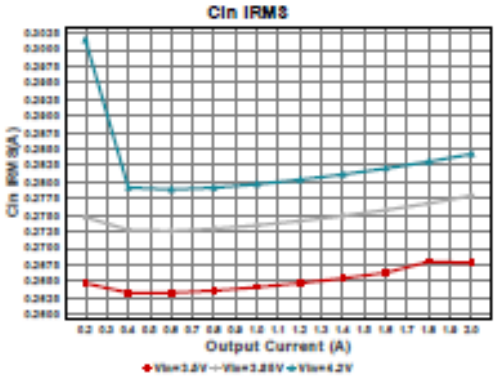
Name	Manufacturer	Part Number	Properties	Qty	Price	Footprint
L1	Bourns	SRR1208-4R5ML	L= 4.5 µH 14.0 mOhm	1	$0.45	 SRR1208 216 mm²
M1	Texas Instruments	CSD13202Q2	VdsMax= 12.0 V IdsMax= 22.0 Amps	1	$0.12	 DQH0006C 9 mm²
Rcomp	Vishay-Dale	CRCW04021K58FKED Series= CRCW..e3	Res= 1.58 kOhm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	 0402 3 mm²
Rfadj	Vishay-Dale	CRCW040226K7FKED Series= CRCW..e3	Res= 26.7 kOhm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	 0402 3 mm²
Rfb1	Vishay-Dale	CRCW04021K00FKED Series= CRCW..e3	Res= 1000.0 Ohm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	 0402 3 mm²
Rfb2	Vishay-Dale	CRCW04026K19FKED Series= CRCW..e3	Res= 6.19 kOhm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	 0402 3 mm²
Rfilt	Vishay-Dale	CRCW0402100RFKED Series= CRCW..e3	Res= 100.0 Ohm Power= 63.0 mW Tolerance= 1.0%	1	$0.01	 0402 3 mm²
Rsense	Stackpole Electronics Inc	CSR1206FK10L0 Series= ?	Res= 10.0 mOhm Power= 500.0 mW Tolerance= 1.0%	1	$0.12	 1206 11 mm²
U1	Texas Instruments	LM3478MM0/NOPB	Switcher	1	$0.73	 MUA08A 24 mm²

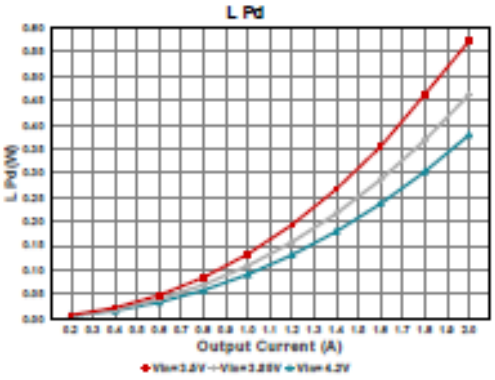
IC TJ

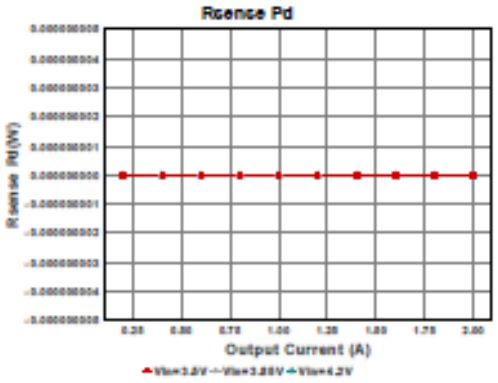


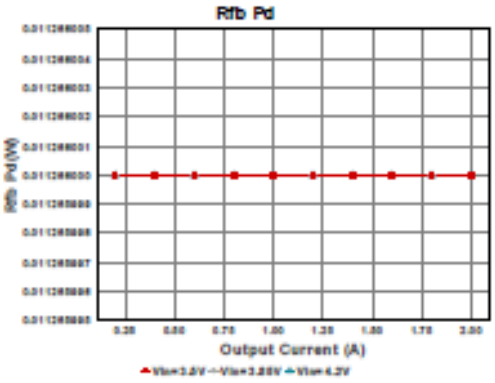
Duty Cycle

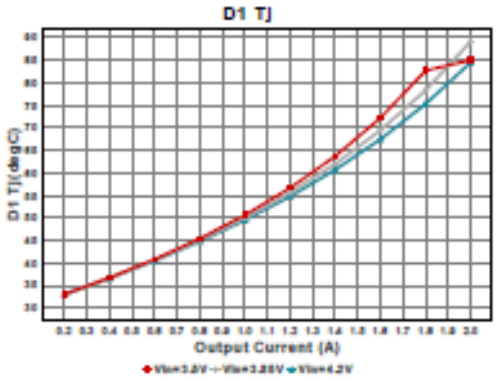


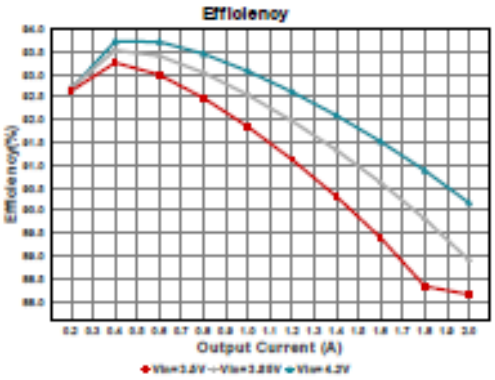


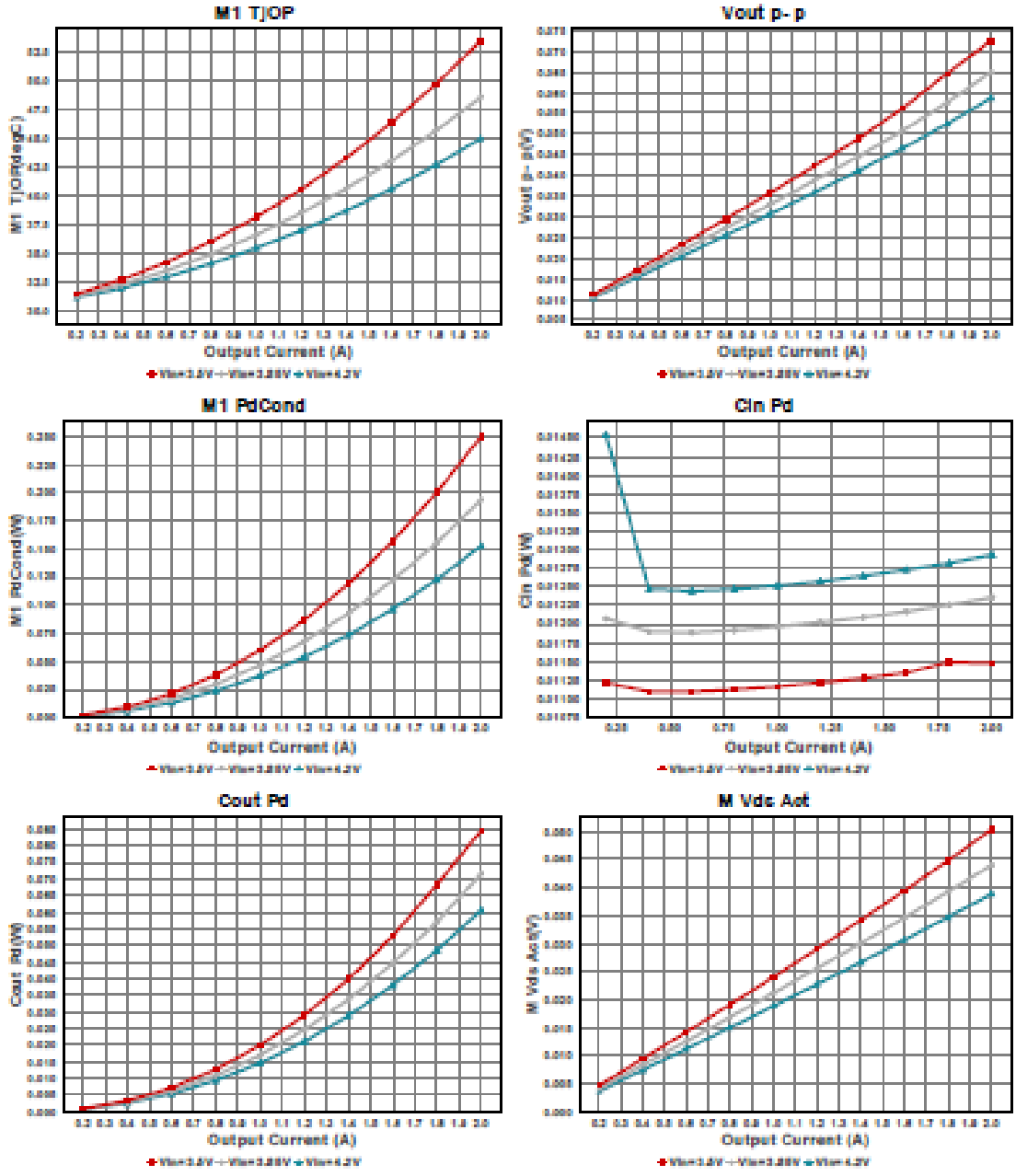


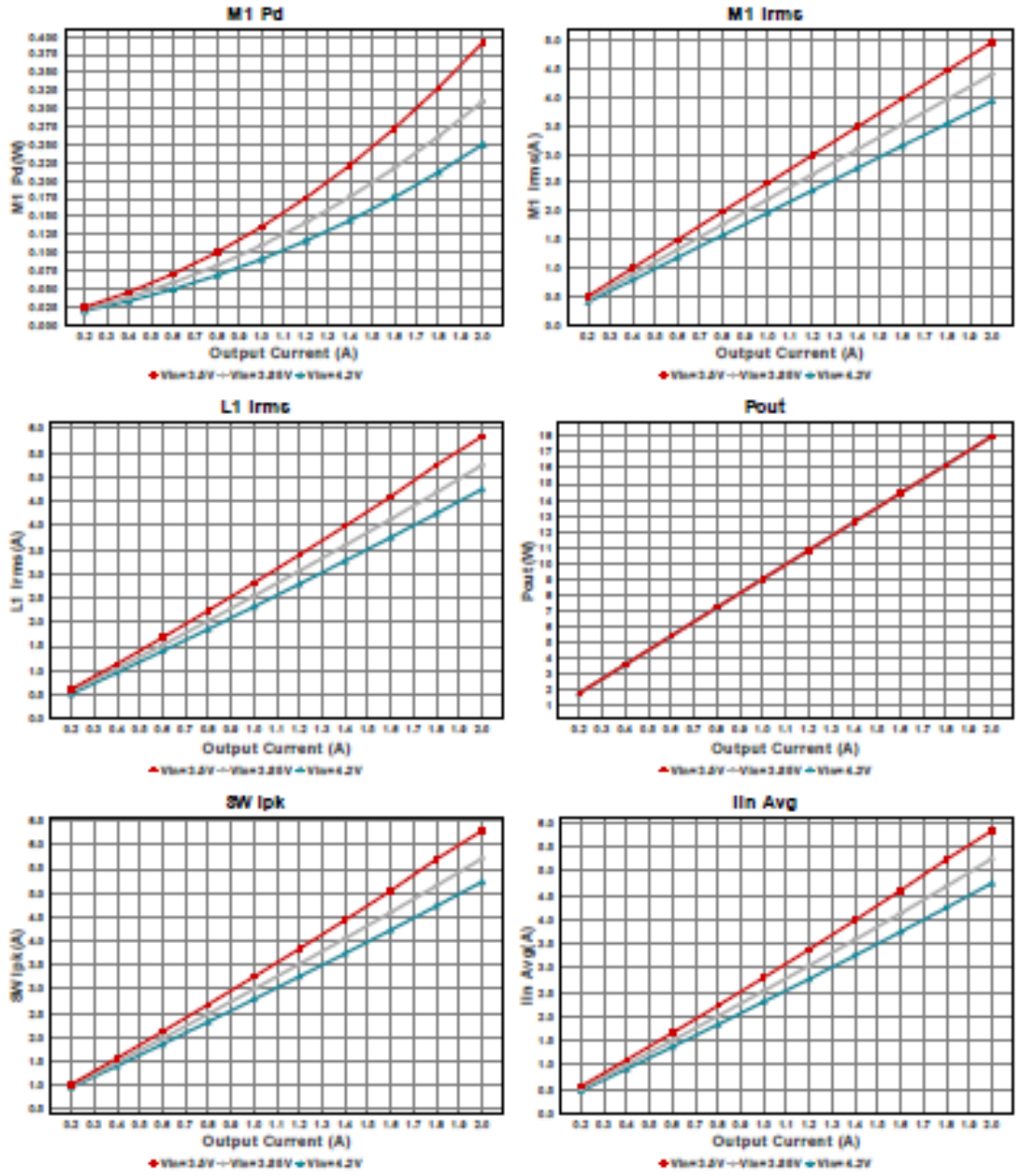


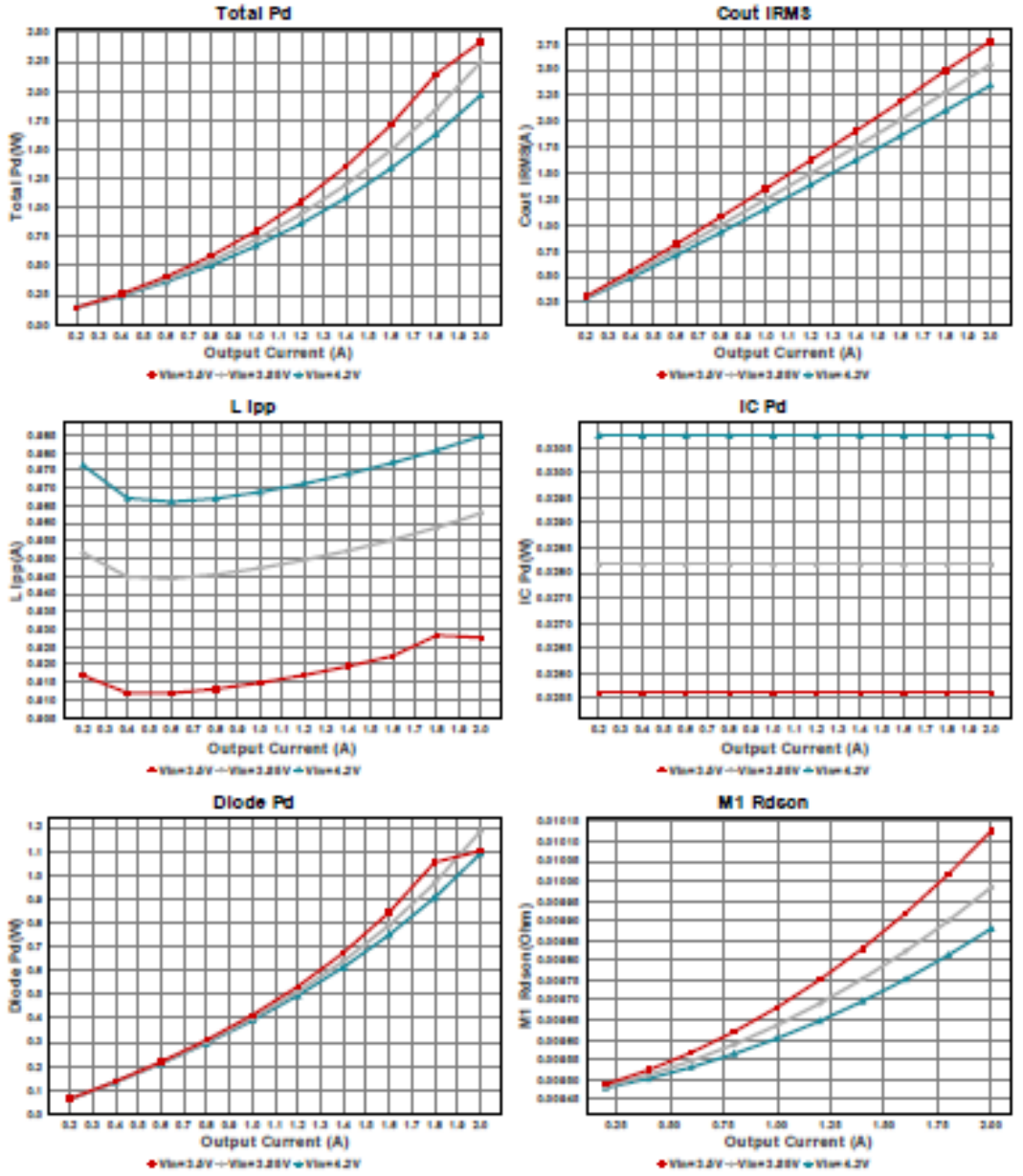


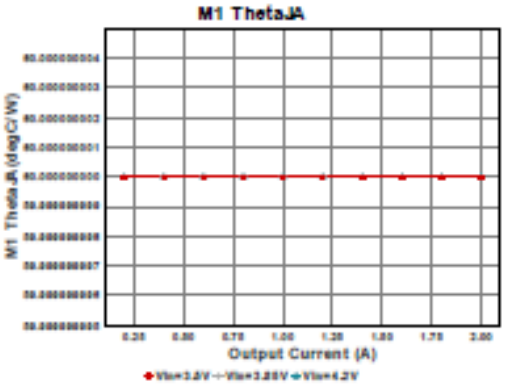


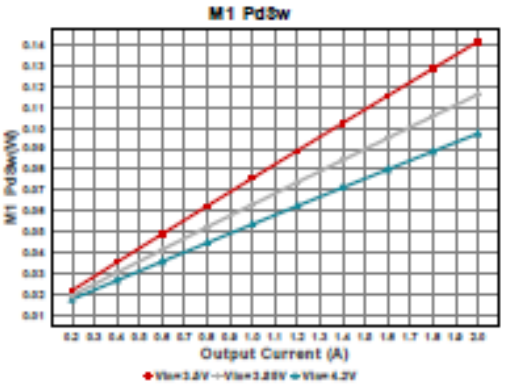


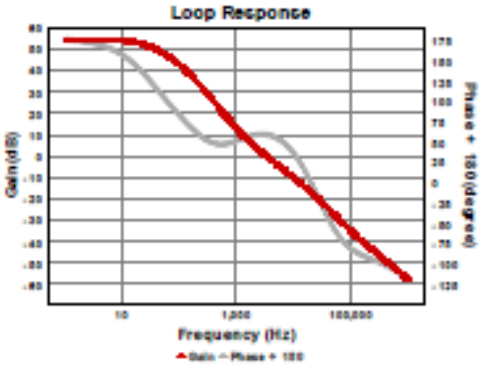

M1 TjOP
M1 TjOP(degC)
Output Current (A)
Vout p-p
Vout p-p(V)
Output Current (A)
M1 PdCond
M1 PdCond(W)
Output Current (A)
Cin Pd
Cin Pd(W)
Output Current (A)
Cout Pd
Cout Pd(W)
Output Current (A)
M Vds Act
M Vds Act(V)
Output Current (A)


M1 Pd
M1 Pd(W)
Output Current (A)
M1 Irms
M1 Irms(A)
Output Current (A)
L1 Irms
L1 Irms(A)
Output Current (A)
Pout
Pout(W)
Output Current (A)
SW Ipk
SW Ipk(A)
Output Current (A)
Iin Avg
Iin Avg(A)
Output Current (A)









Operating Values

#	Name	Value	Category	Description
1.	Cin IRMS	284.312 mA	Capacitor	Input capacitor RMS ripple current
2.	Cin Pd	11.178 mW	Capacitor	Input capacitor power dissipation
3.	Cout IRMS	2.72 A	Capacitor	Output capacitor RMS ripple current
4.	Cout Pd	81.984 mW	Capacitor	Output capacitor power dissipation
5.	D1 Tj	85.0 degC	Diode	D1 junction temperature
6.	Diode Pd	1.1 W	Diode	Diode power dissipation
7.	IC Pd	25.781 mW	IC	IC power dissipation
8.	IC Tj	35.156 degC	IC	IC junction temperature
9.	IC Tolerance	24.3 mV	IC	IC Feedback Tolerance
10.	ICThetaJA	200.0 degC/W	IC	IC junction-to-ambient thermal resistance
11.	Iin Avg	5.71 A	IC	Average input current
12.	L Ipp	915.6 mA	Inductor	Peak-to-peak inductor ripple current
13.	L Pd	544.34 mW	Inductor	Inductor power dissipation
14.	L1 Irms	5.692 A	Inductor	Inductor ripple current
15.	M Vds Act	50.339 mV	Mosfet	M Vds
16.	M1 Irms	4.069 A	Mosfet	M1 MOSFET Irms
17.	M1 Pd	393.79 mW	Mosfet	M1 MOSFET total power dissipation
18.	M1 PdCond	250.13 mW	Mosfet	M1 MOSFET conduction losses
19.	M1 PdSw	143.66 mW	Mosfet	M1 MOSFET switching losses
20.	M1 Rdson	10.131 mOhm	Mosfet	Drain-Source On-resistance
21.	M1 ThetaJA	60.0 degC/W	Mosfet	MOSFET junction-to-ambient thermal resistance
22.	M1 TjOP	53.628 degC	Mosfet	M1 MOSFET junction temperature
23.	Cin Pd	11.178 mW	Power	Input capacitor power dissipation
24.	Cout Pd	81.984 mW	Power	Output capacitor power dissipation
25.	Diode Pd	1.1 W	Power	Diode power dissipation
26.	IC Pd	25.781 mW	Power	IC power dissipation
27.	L Pd	544.34 mW	Power	Inductor power dissipation
28.	M1 Pd	393.79 mW	Power	M1 MOSFET total power dissipation
29.	M1 PdCond	250.13 mW	Power	M1 MOSFET conduction losses
30.	M1 PdSw	143.66 mW	Power	M1 MOSFET switching losses
31.	Rfb Pd	11.266 mW	Power	Rfb Power Dissipation
32.	Rsense Pd	297.8 mW	Power	LED Current Rsns Power Dissipation

#	Name	Value	Category	Description
33.	Total Pd	1.084 W	Power	Total Power Dissipation
34.	Rfb Pd	11.266 mW	Resistor	Rfb Power Dissipation
35.	Rsense Pd	297.8 mW	Resistor	LED Current Rsns Power Dissipation
36.	BOM Count	18	System Information	Total Design BOM count
37.	Cross Freq	2.835 kHz	System Information	Bode plot crossover frequency
38.	Duty Cycle	64.826 %	System Information	Duty cycle
39.	Efficiency	90.074 %	System Information	Steady state efficiency
40.	FootPrint	579.0 mm²	System Information	Total Foot Print Area of BOM components
41.	Frequency	541.734 kHz	System Information	Switching frequency
42.	Gain Marg	-16.043 dB	System Information	Bode Plot Gain Margin
43.	Iout	2.0 A	System Information	Iout operating point
44.	Low Freq Gain	52.926 dB	System Information	Gain at 1Hz
45.	Mode	CCM	System Information	Conduction Mode
46.	Phase Marg	58.63 deg	System Information	Bode Plot Phase Margin
47.	Pout	18.0 W	System Information	Total output power
48.	SW Ipk	6.144 A	System Information	Peak switch current
49.	Total BOM	$2.4	System Information	Total BOM Cost
50.	Vin	3.5 V	System Information	Vin operating point
51.	Vout	9.0 V	System Information	Operational Output Voltage
52.	Vout Actual	9.059 V	System Information	Vout Actual calculated based on selected voltage divider resistors
53.	Vout Tolerance	3.701 %	System Information	Vout Tolerance based on IC Tolerance (no load) and voltage divider resistors if applicable
54.	Vout p-p	70.807 mV	System Information	Peak-to-peak output ripple voltage

Design Inputs

Name	Value	Description
Iout	2.0	Maximum Output Current
VinMax	4.2	Maximum input voltage
VinMin	3.5	Minimum input voltage
Vout	9.0	Output Voltage
base_pn	LM3478	Base Product Number
source	DC	Input Source Type
Ta	30.0	Ambient temperature

WEBENCH® Assembly

Component Testing

Some published data on components in datasheets such as Capacitor ESR and Inductor DC resistance is based on conservative values that will guarantee that the components always exceed the specification. For design purposes it is usually better to work with typical values. Since this data is not always available it is a good practice to measure the Capacitance and ESR values of Cin and Cout, and the inductance and DC resistance of L1 before assembly of the board. Any large discrepancies in values should be electrically simulated in WEBENCH to check for instabilities and thermally simulated in WebTHERM to make sure critical temperatures are not exceeded.

Soldering Component to Board

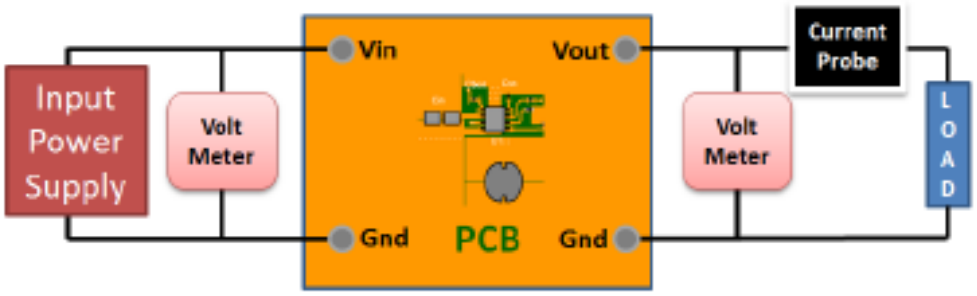
If board assembly is done in house it is best to tack down one terminal of a component on the board then solder the other terminal. For surface mount parts with large tabs, such as the DPAK, the tab on the back of the package should be pre-tinned with solder, then tacked into place by one of the pins. To solder the tab down to the board place the iron down on the board while resting against the tab, heating both surfaces simultaneously. Apply light pressure to the top of the plastic case until the solder flows around the part and the part is flush with the PCB. If the solder is not flowing around the board you may need a higher wattage iron (generally 25W to 30W is enough).

Initial Startup of Circuit

It is best to initially power up the board by setting the input supply voltage to the lowest operating input voltage 3.5V and set the input supply's current limit to zero. With the input supply off connect up the input supply to Vin and GND. Connect a digital volt meter and a load if needed to set the minimum Iout of the design from Vout and GND. Turn on the input supply and slowly turn up the current limit on the input supply. If the voltage starts to rise on the input supply continue increasing the input supply current limit while watching the output voltage. If the current increases on the input supply, but the voltage remains near zero, then there may be a short or a component misplaced on the board. Power down the board and visually inspect for solder bridges and recheck the diode and capacitor polarities. Once the power supply circuit is operational then more extensive testing may include full load testing, transient load and line tests to compare with simulation results.

Load Testing

The setup is the same as the initial startup, except that an additional digital voltmeter is connected between Vin and GND, a load is connected between Vout and GND and a current meter is connected in series between Vout and the load. The load must be able to handle at least rated output power + 50% (7.5 watts for this design). Ideally the load is supplied in the form of a variable load test unit. It can also be done in the form of suitably large power resistors. When using an oscilloscope to measure waveforms on the prototype board, the ground leads of the oscilloscope probes should be as short as possible and the area of the loop formed by the ground lead should be kept to a minimum. This will help reduce ground lead inductance and eliminate EMI noise that is not actually present in the circuit.

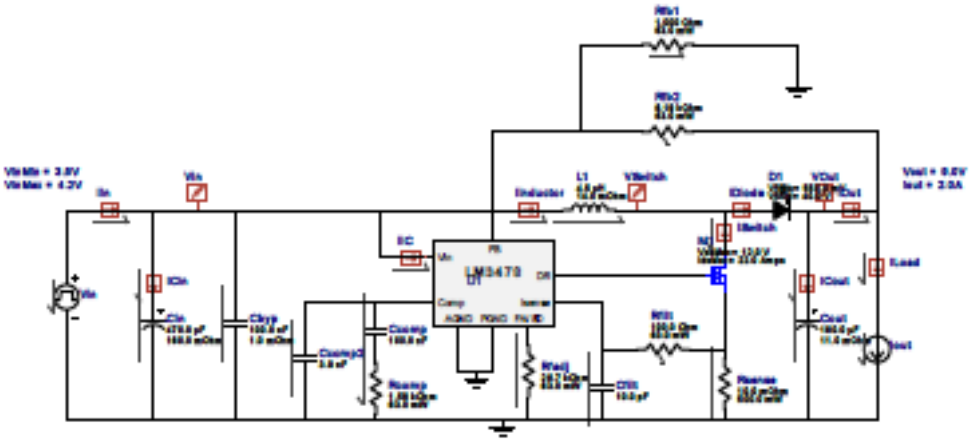


WEBENCH® Electrical Simulation Report

Design Id = 4

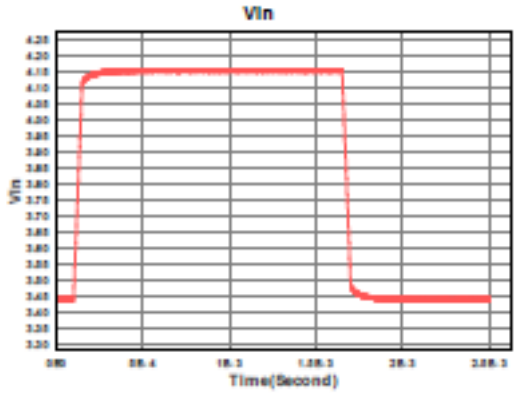
sim_id = 1

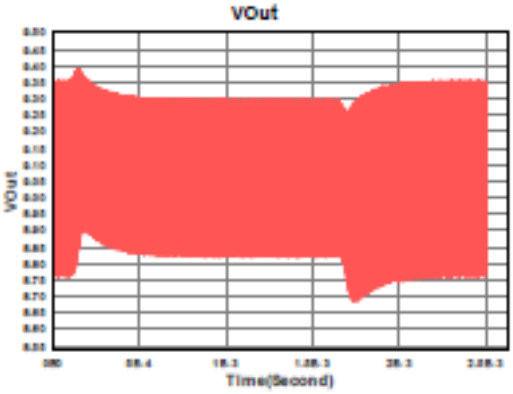
Simulation Type = Input Transient



Simulation Parameters

#	Name	Parameter Name	Description	Values
1.	Iout	I	Load Current	2.0 A





Design Assistance

1. Master key : E51B955EA55D6907[v1]
2. LM3478 Product Folder : http://www.ti.com/product/LM3478 : contains the data sheet and other resources.

Annexe 2(Schéma électrique) :

Figure 2

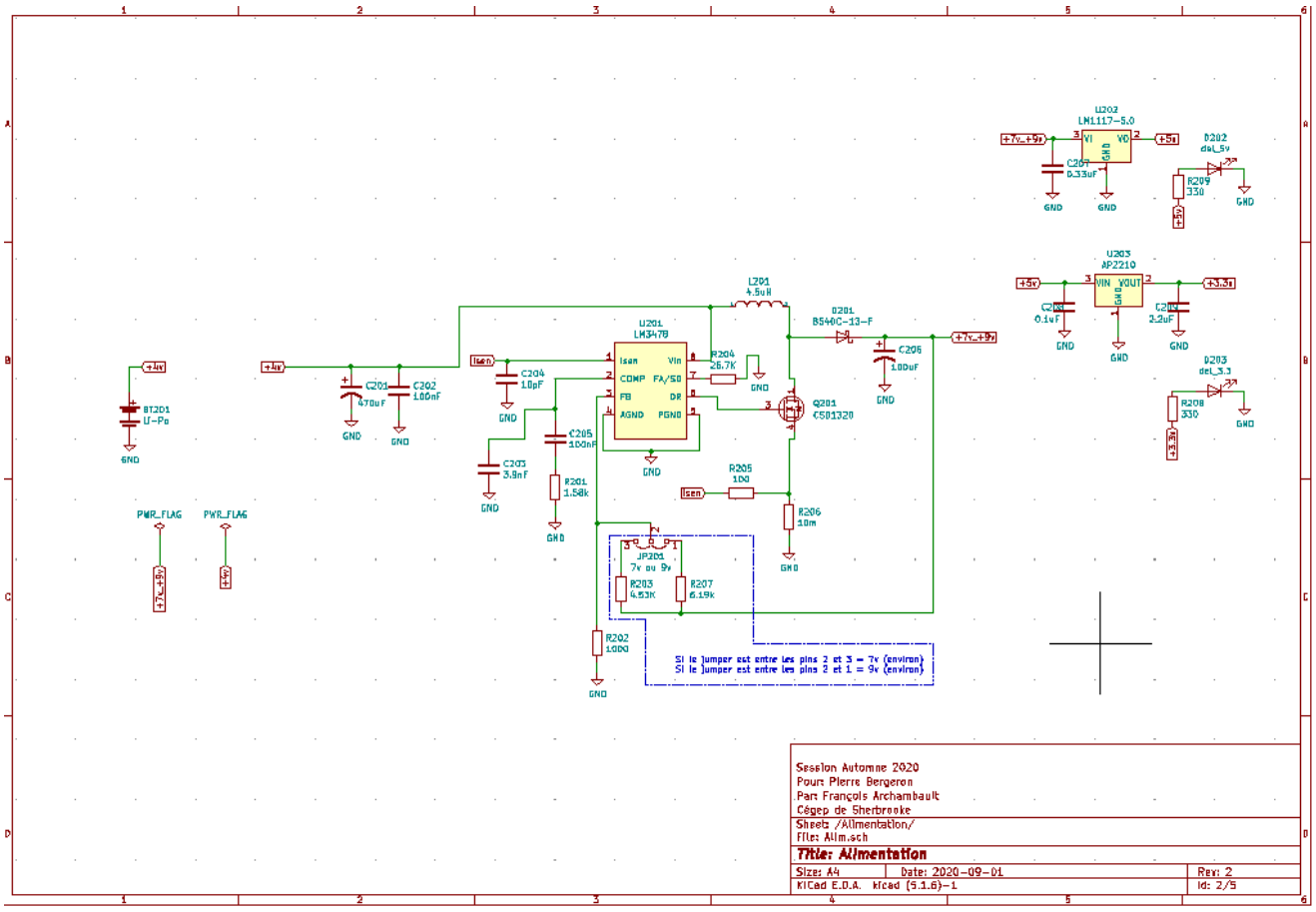


Figure 3

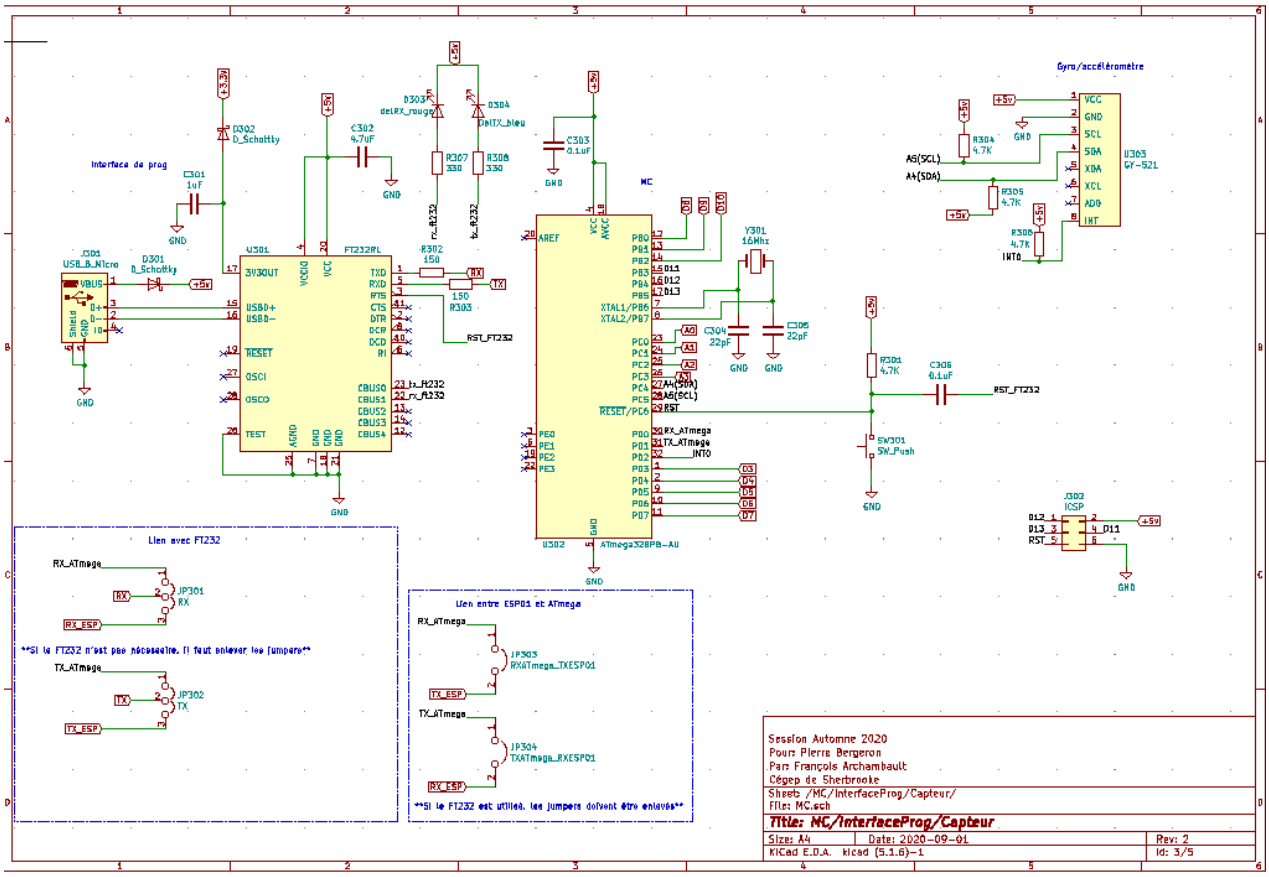

Interface de prog
Gyro/accéléromètre
Lien avec FT232
Lien entre ESP01 et ATmega
Si le FT232 n'est pas nécessaire, il faut enlever les jumpers
Si le FT232 est utilisé, les jumpers doivent être enlevés
Session Automne 2020
Pour: Pierre Bergeron
Par: François Archambault
Cégep de Sherbrooke
Sheet: /MC/InterfaceProg/Capteur/
File: MC.sch
Title: MC/InterfaceProg/Capteur
Size: A4
Date: 2020-09-01
Rev: 2
KiCad E.D.A. kicad (5.1.6)-1
Id: 3/5

Figure 4

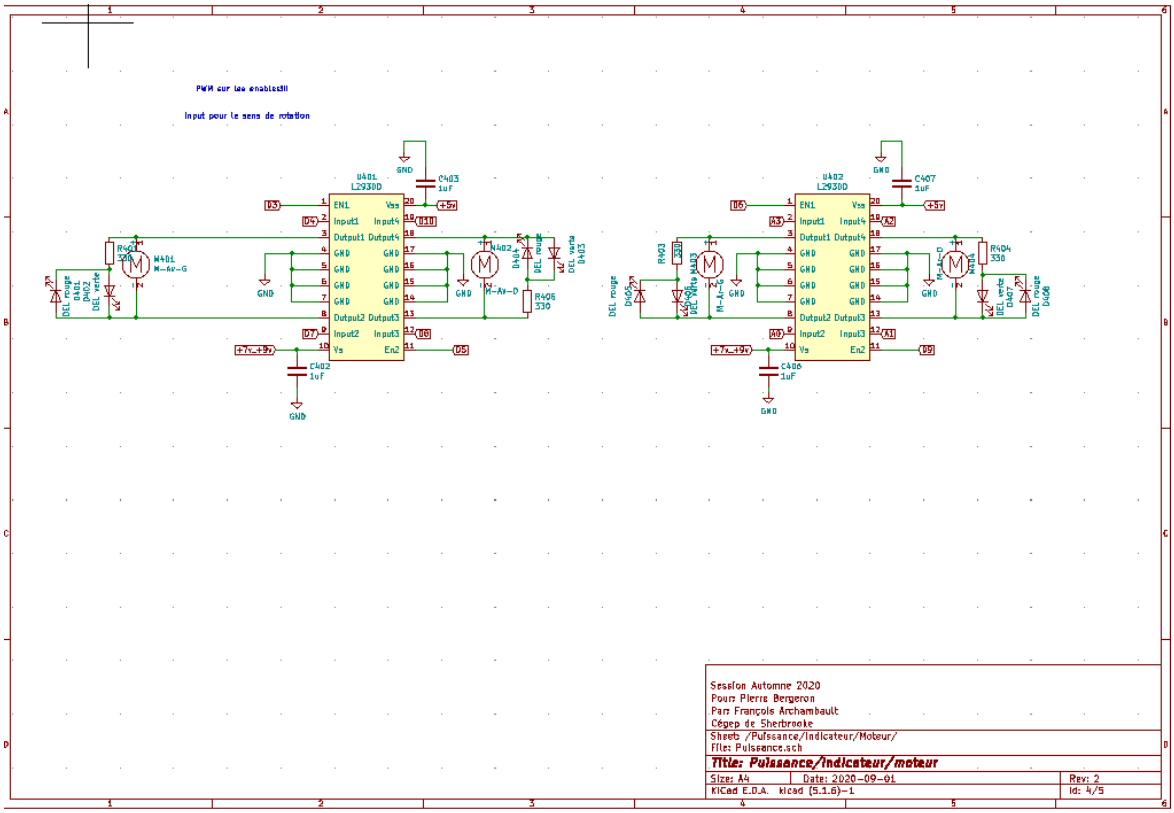
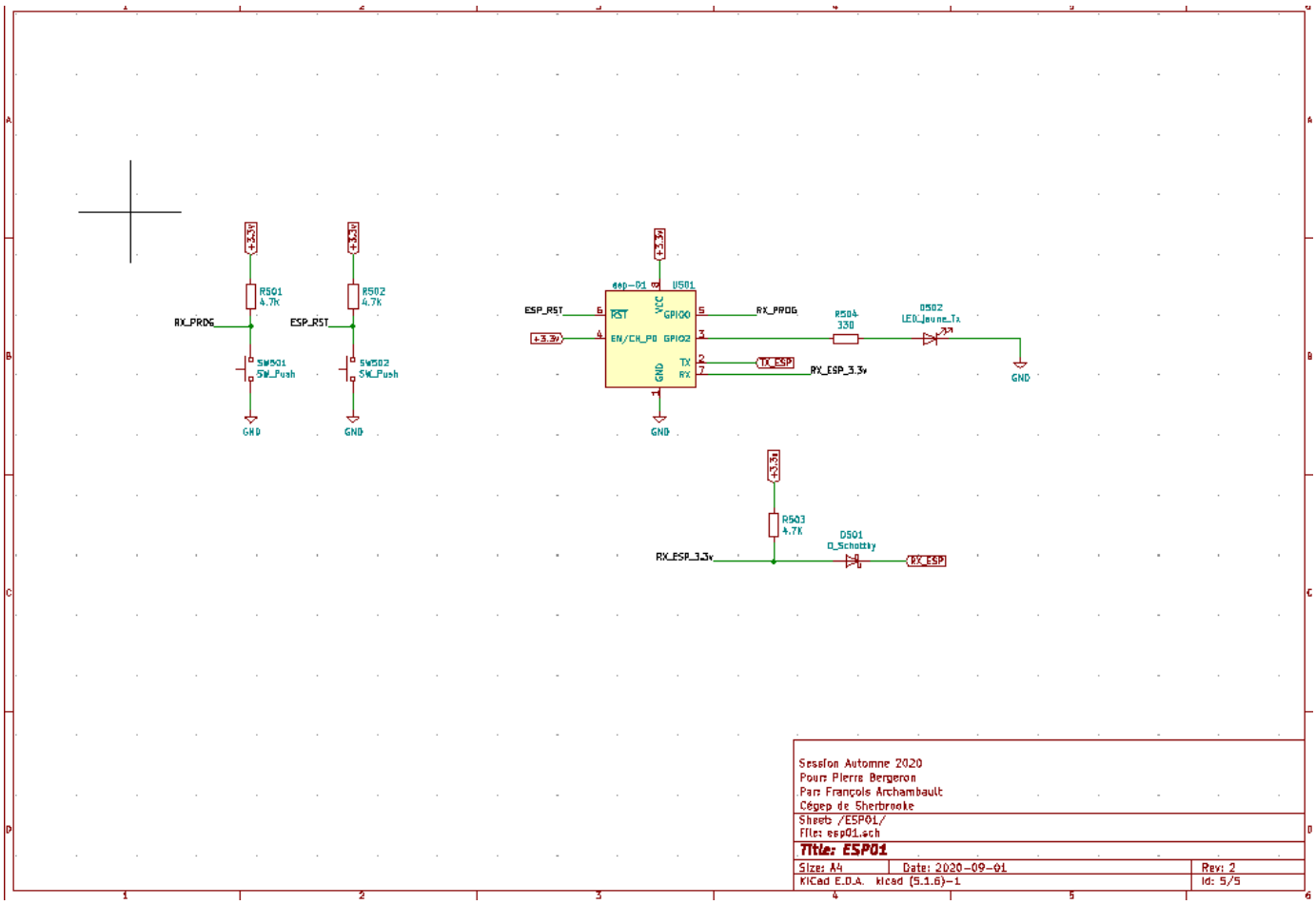
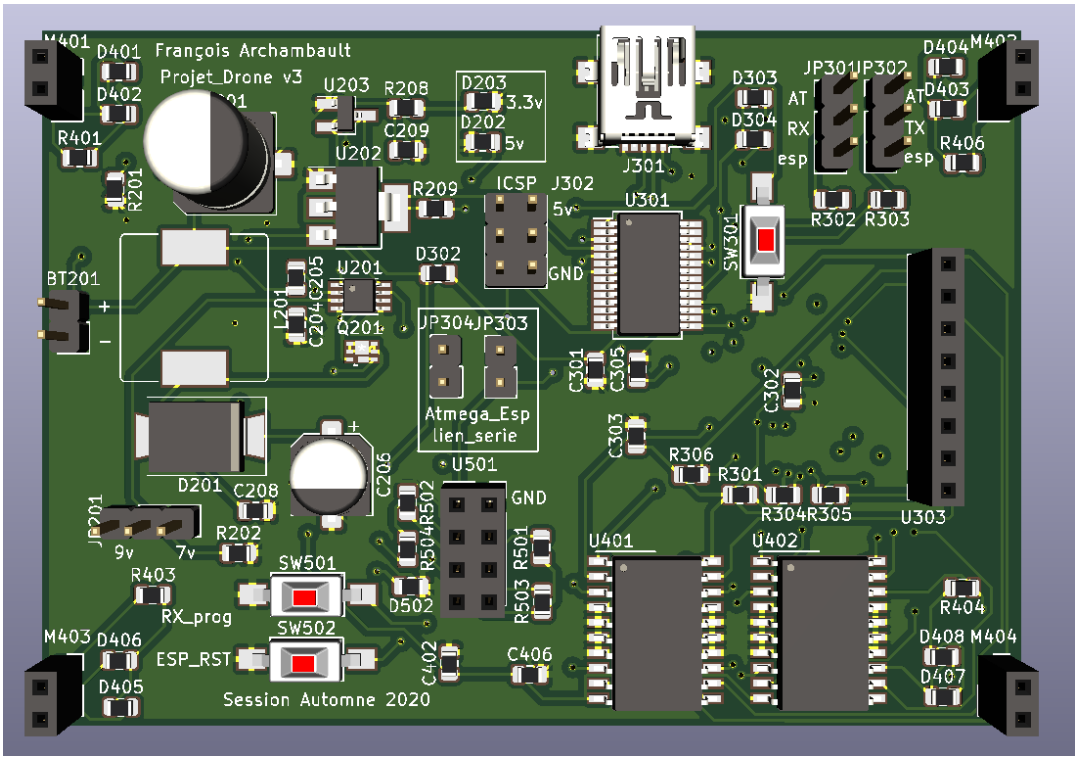

PWM sur les enables!!!
Input pour le sens de rotation
U401
L2930D
U402
L2930D
C403
1uF
C407
1uF
C402
1uF
C406
1uF
GND
+5v
+7v_+9v
M401
M-Av-G
M402
M-Av-D
M403
M-Ar-G
M404
M-Ar-D
R401
330
R402
330
R403
330
R404
330
R405
330
R406
330
DEL rouge
DEL verte
Sessfon Automne 2020
Pour Pierre Bergeron
Par François Archambault
Cégep de Sherbrooke
Sheet: /Puissance/Indicateur/Moteur/
File: Puissance.sch
Title: Puissance/Indicateur/moteur
Size: A4
Date: 2020-09-01
Rev: 2
KiCad E.D.A. kicad (5.1.6)-1
Id: 4/5

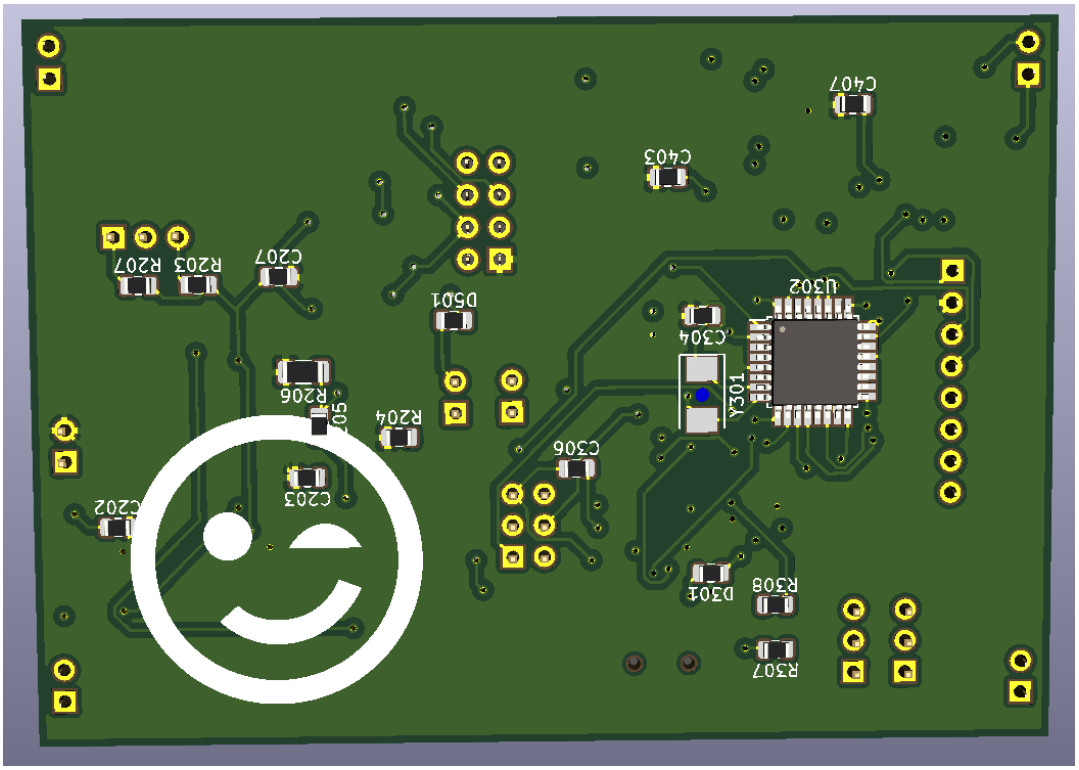
Figure 5


+3.3V
R501
4.7K
R502
4.7K
RX_PROG
ESP_RST
SW501
SW_Push
SW502
SW_Push
GND
esp-01
U501
RST
VCC
GPIO0
EN/CH_PD
GPIO2
TX
RX
R504
330
D502
LED_Jaune_Tx
TX_ESP
RX_ESP_3.3v
R503
4.7K
D501
D_Schottky
RX_ESP
Session Automne 2020
Pour Pierre Bergeron
Par François Archambault
Cégep de Sherbrooke
Sheet: /ESP01/
File: esp01.sch
Title: ESP01
Size: A4
Date: 2020-09-01
Rev: 2
KiCad E.D.A. kicad (5.1.6)-1
Id: 5/5

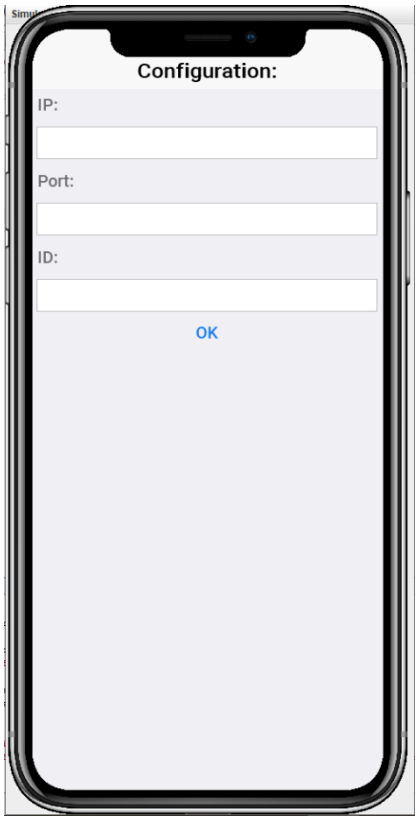
Annexe 3 (vue 3D):

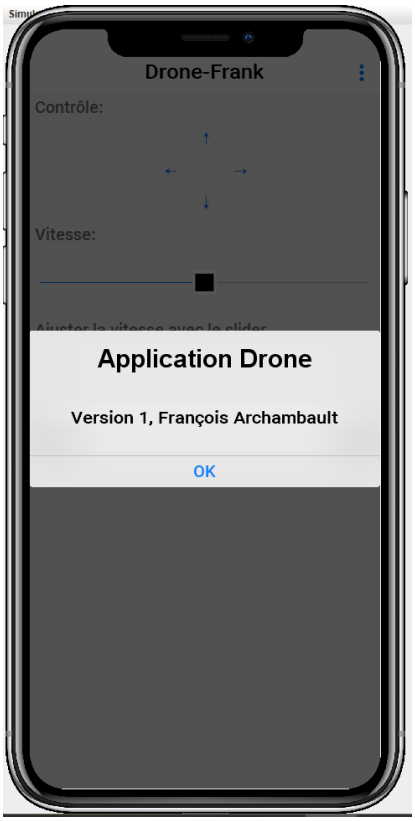
Vues 3D PCB version 2





Annexe 4 (Apparence des fenêtres de l'application):





Annexe 5 (CodeNameOne):

```
/*
* Programme de base pour le fonctionnement de l'application java.
* Par François Archambault
* Pour Pierre Bergeron
* Session Automne 2020
*/


package tso.cegepSherbrooke.droneFA;


import static com.codename1.ui.CN.*;
import com.codename1.ui.Display;
import com.codename1.ui.Form;
import com.codename1.ui.Dialog;
import com.codename1.ui.Label;
import com.codename1.ui.plaf.UIManager;
import com.codename1.ui.util.Resources;
import com.codename1.io.Log;
import com.codename1.ui.Toolbar;
import java.io.IOException;
import com.codename1.ui.layouts.BoxLayout;
import com.codename1.io.NetworkEvent;
import com.codename1.ui.Component;
import com.codename1.ui.layouts.FlowLayout;

/**
```

```
 * This file was generated by <a href="https://www.codenameone.com/">Codename One</a> for the purpose
 * of building native mobile applications using Java.
 */
public class monApp {

    private Form current;
    private Resources theme;

    public void init(Object context) {
        // use two network threads instead of one
        updateNetworkThreadCount(2);

        theme = UIManager.initFirstTheme("/theme");

        // Enable Toolbar on all Forms by default
        Toolbar.setGlobalToolbar(true);

        // Pro only feature
        Log.bindCrashProtection(true);

        addNetworkErrorListener(err -> {
            // prevent the event from propagating
            err.consume();
            if(err.getError() != null) {
                Log.e(err.getError());
            }
            Log.sendLogAsync();
```

```
        Dialog.show("Connection Error", "There was a networking error in the connection to " +
err.getConnectionRequest().getUrl(), "OK", null);
      });
   }

   public void start() {
      if(current != null){
         current.show();
         return;
      }
      affichage aff = new affichage(); //appel à la classe affiche qui génère l'application
   }

   public void stop() {
      current = getCurrentForm();
      if(current instanceof Dialog) {
         ((Dialog)current).dispose();
         current = getCurrentForm();
      }
   }
   public void destroy() {
   }
}
```

```
/*
 * Fichier pour la section affiche de la fenêtre principale de l'application (classe affichage)
 * Le programme gère aussi l'envoie des commandes et la résception des valeurs de l'accéléromètre par TCP
 * Affiche les données de l'accéléromètre
 * Par François Archambault
 * Session Automne 2020
 */
package tso.cegepSherbrooke.droneFA; //doit avoir le même package que les autre fichier du programme


import static com.codename1.io.JSONParser.parse;
import static com.codename1.ui.CN.*;
import com.codename1.ui.Display;
import com.codename1.ui.Form;
import com.codename1.ui.Dialog;
import com.codename1.ui.Label;
import com.codename1.ui.plaf.UIManager;
import com.codename1.ui.util.Resources;
import com.codename1.io.Log;
import com.codename1.ui.Toolbar;
import java.io.IOException;
import com.codename1.ui.layouts.BoxLayout;
import com.codename1.io.NetworkEvent;
import com.codename1.io.Socket;
import com.codename1.io.SocketConnection;
import com.codename1.ui.Button;
import com.codename1.ui.Command;
```

```
import com.codename1.ui.Component;
import com.codename1.ui.Container;
import com.codename1.ui.Font;
import com.codename1.ui.FontImage;
import com.codename1.ui.Image;
import com.codename1.ui.Slider;
import com.codename1.ui.TextField;
import com.codename1.ui.events.ActionEvent;
import com.codename1.ui.layouts.BorderLayout;
import com.codename1.ui.layouts.FlowLayout;
import com.codename1.ui.layouts.GridLayout;
import com.codename1.ui.plaf.Border;
import com.codename1.ui.plaf.Style;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import com.codename1.io.Util;

/**
 *
 * @author farch
 */
public class affichage extends Form
{

    public Command cAProposDe; //onglet dans menu
    public Command cQuitter; //onglet dans menu
    public Command cConfig; //onglet dans menu
    public String IP = ""; //variable pour l'IP
```

```
   public String PORT = ""; //variable pour le PORT
   public String ID = ""; //variable pour l'ID

   Label valueX = new Label("X=0"); //Labels qui vont accueillir les données de l'accéléramètre
   Label valueY = new Label("Y=0"); //ne sont pas dans le constructeur parce que la méthode
send a besoin de changer le texte des labels
   Label valueZ = new Label("Z=0");

   public affichage() //constructeur de la classe
   {

      setTitle("Drone-Frank"); //titre de l'application
      setLayout( BoxLayout.y()); //layout de la fenêtre en box (bloc un par dessus l'autre)
      Container panelControle = new Container(); //container
      Container pGrid = new Container(); //

      //instanciation de tous les labels utiles dans la fenêtre
      Label lb = new Label("Contrôle: ");
      Label vitesse = new Label("Vitesse: ");
      Label value = new Label("Ajuster la vitesse avec le slider...");
      Label accelerometre = new Label("Accéléromètre: ");

      //instanciation des boutons utiles dans la fenêtre
      Button bHaut = new Button("↑");
      Button bBas = new Button("↓");
      Button bGauche = new Button("←");
      Button bDroite = new Button("→");
```

```
    //Attribut une action au bouton. Ici c'est un appel à la méthode send avec une direction en paramètre
    bHaut.addActionListener((e) -> send("Front*"));
    bBas.addActionListener((e) -> send("Back*"));
    bGauche.addActionListener((e) -> send("Gauche*"));
    bDroite.addActionListener((e) -> send("Droite*"));



    //section du slider
    Style thumbStyle = new Style(); //pour mettre un curseur sur le slider
    thumbStyle.setFont(Font.createSystemFont(Font.FACE_MONOSPACE, Font.STYLE_BOLD, Font.SIZE_LARGE), true);


    Slider sliderVitesse = new Slider(); //instanciation
    sliderVitesse.setEditable(true); //le slider peut changer de valeur
    sliderVitesse.setMinValue(0); //MIN =0
    sliderVitesse.setMaxValue(100); //MAX = 100
    sliderVitesse.setProgress(50); //commence à 50 (milieu)
    sliderVitesse.setThumbImage(FontImage.create("■", thumbStyle)); //le curseur est un carré
    sliderVitesse.addActionListener(e -> {       //attribut une action quand le slider change de valeur
            Integer valueSelected = sliderVitesse.getProgress(); //valeur actuelle du slider (entre 0 et 100)
            value.setText( valueSelected.toString() + "% de vitesse"); //change le texte de la vitesse pour afficher la vitesse (ex. 43% de vitesse)
            String toSend = valueSelected.toString()+ "/"; //création de la string qui va être envoyée à l'esp01 par TCP
            send(toSend); //appel la méthode send avec en paramètre la vitesse + / (ex. 34/)
});
```

```
        //instanciation de labels vides pour remplir la grille pour faire le layout (nord, sud, est,
ouest)
        Label v1 = new Label("");
        Label v2 = new Label("");
        Label v3 = new Label("");
        Label v4 = new Label("");
        Label v5 = new Label("");
        pGrid.setLayout(new GridLayout(3,3)); //grille de 3 par 3
        panelControle.setLayout(new FlowLayout(Component.CENTER));

        //ajoute à la grille tous les labels et les boutons pour faire le bon layout des directions
        pGrid.add(v1);
        pGrid.add(bHaut);
        pGrid.add(v2);
        pGrid.add(bGauche);
        pGrid.add(v3);
        pGrid.add(bDroite);
        pGrid.add(v4);
        pGrid.add(bBas);
        pGrid.add(v5);


        panelControle.add(pGrid); //ajoute la grille au panel


        //ajoute tout à la fenêtre principale dans le bon ordre. (lb va être le premier en haut de la
fenêtre et valuez le dernier)
```

```
add(lb);
add(panelControle);
add(vitesse);
add(sliderVitesse);
add(value);
add(accelerometre);
add(valueX);
add(valueY);
add(valueZ);

//onglet a propos de dans le menu
cAProposDe = new Command("À propos de...")
{
   public void actionPerformed(ActionEvent ae) //quand appuyé
   {
      onAProposDe(); //appel à la méthode onAProposDe
   }
};
//onglet Quitter de dans le menu
cQuitter = new Command("Quitter")
{
   public void actionPerformed(ActionEvent ae)//quand appuyé
   {
      onQuitter();//appel à la méthode onQuitter
   }
};
//onglet configuration de dans le menu
cConfig = new Command("Configuration")
{
```

```
        public void actionPerformed(ActionEvent ae)//quand appuyé
        {
            onConfig();//appel à la méthode onQuitter
        }
    };


    Toolbar tb = getToolbar(); //nouveau menu
    tb.addCommandToOverflowMenu(cConfig); //ajoute les onglets au menu
    tb.addCommandToOverflowMenu(cAProposDe);
    tb.addCommandToOverflowMenu(cQuitter);


    show();     //affiche les composantes de la fenêtre
  }
  public void onConfig()
  {
    new config(this); //appel à la classe config (autre fenêtre qui sert à l'affichage de la fenêtre de config
  }


  public void onQuitter()
  {
    System.exit(0); //quitte l'application
  }
  public void onAProposDe()
  {
    Dialog.show("Application Drone", "Version 1, François Archambault", "OK", null); //affiche un message
  }
```

```
public void send(String bufferEnvoie) //méthode qui envoie une commande à l'esp01 par TCP
{
   try
   {
      // Ouverture de la connexion au serveur sur le port 8888 (adresse localhost OK avec simulateur mais pas avec vrai mobile)
      Socket.connect(IP, Integer.parseInt(PORT), new SocketConnection()
      {
         // Surcharge de la méthode pour la gestion des erreurs de connexion

         public void connectionError(int errorCode, String message)
         {
            System.out.println("Error");
         }

          // Surcharge de la méthode lorsque la connexion est établie

         public void connectionEstablished(InputStream is, OutputStream os) {
            try {
               if(isConnected()) {

                  os.write(bufferEnvoie.getBytes());  // Envoi le buffer de caractères contenant la commande
                  try
                  {
                     Thread.sleep(100);  // Délai pour recevoir la réponse...
                  }
                  catch(Exception e)
```

```
                {
                   System.out.println(e.toString());
                }
                InputStreamReader isr = new InputStreamReader(is); // Définir la source de donnée et le canal de communication
                String inputLine;
                StringBuffer response = new StringBuffer();
                int nb = 0;
                do
                {
                   char[] cbuff = new char[1000];
                   nb=isr.read(cbuff, 0, 100);   // Lecture de la réponse en cours (peut-être contenu dans plusieurs trames TCP)
                   inputLine = new String(cbuff);
                   response.append(inputLine);
                }while (is.available()>0);
                isr.close();  // Fermeture de la connexion

                //Affiche la réponse reçue
                System.out.println(response.toString());
                String[] valAcc1 = Util.split(response.toString(),"/"); //divise la réponse au "/" et c'Est mis dans un tableau de string

                String[] valAcc2 = Util.split(valAcc1[0].toString()," ");//divise le tableau valAcc1 à l'index 0 au " " et c'Est mis dans un autre tableau de string

                int longueurAcc = valAcc2[5].length(); //prend la longueur de l'ID (valAcc2[5])
                int longueurID = ID.length();
                //if(valAcc1[5].equals(ID))
```

```
                    if(longueurAcc == longueurID) //compare la longueur de l'ID et de l'ID reçu par
TCP. SI pareil
                    {
                       valueX.setText("X: " + valAcc2[2].toString()); //change les valeurs de x y z par
les nouvelles
                       valueY.setText("Y: " + valAcc2[3].toString());
                       valueZ.setText("Z: " + valAcc2[4].toString());
                      // ID = valAcc[3];
                    }
                    for(int i=0; i< valAcc2.length;i++) //vide le tableau
                    {
                       //valAcc1[i]="";
                       valAcc2[i]="";
                    }



                  }
               } catch(Exception err) {
                  err.printStackTrace();
               }
            }
         });
      }
      catch(Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

```
/*
 * Fichier de la fenêtre de configuration. Permet de changer les variables de connexion de l'application.
 * Par François Archambault
 * Session Automne 2020
 */
package tso.cegepSherbrooke.droneFA;

import static com.codename1.ui.CN.*;
import com.codename1.ui.Display;
import com.codename1.ui.Form;
import com.codename1.ui.Dialog;
import com.codename1.ui.Label;
import com.codename1.ui.plaf.UIManager;
import com.codename1.ui.util.Resources;
import com.codename1.io.Log;
import com.codename1.ui.Toolbar;
import java.io.IOException;
import com.codename1.ui.layouts.BoxLayout;
import com.codename1.io.NetworkEvent;
import com.codename1.ui.Button;
import com.codename1.ui.Component;
import com.codename1.ui.Container;
import com.codename1.ui.TextField;
import com.codename1.ui.events.ActionEvent;
import com.codename1.ui.events.ActionListener;
import com.codename1.ui.layouts.FlowLayout;
import com.codename1.ui.plaf.Border;
import com.codename1.ui.plaf.Style;
```

```
public class config extends Form //classe qui hérite de Form
{
   public affichage m_sParent; //objet de classe affichage
   public TextField tf_IP = new TextField(); //3 textFields pour IP, PORT, ID
   public TextField tf_PORT = new TextField();
   public TextField tf_ID = new TextField();

   public config(affichage sParent) //constructeur
   {

      m_sParent = sParent; //permet de revenir à la fenêtre principale
      setTitle("Configuration:"); //titre
      setLayout(BoxLayout.y()); //création du layout de la fenêtre
      add(new Label("IP: " ));
      add(tf_IP);
      add(new Label("Port: " ));
      add(tf_PORT);
      add(new Label("ID: " ));
      add(tf_ID);



      Button bOK = new Button("OK"); //instanciation d'un bouton OK
      bOK.addActionListener((e) -> btnOK());  //quand bouton OK est appuyé
      bOK.addActionListener((e) -> m_sParent.showBack());  //retourne à la fenêtre principale

      add(bOK); //ajoute le bouton à la fenêtre config
      show(); //affiche les composantes
   }
```

```
    public void btnOK() //méthode appelée quand le bouton OK est appuyé
    {
        m_sParent.IP = tf_IP.getText(); //prend les valeurs écrites dans les textFields pour changer les variables de connexion dans la fenêtre principale
        m_sParent.PORT = tf_PORT.getText();
        m_sParent.ID = tf_ID.getText();

    }

}
```

Annexe 6 (esp01):

```
/*
 * Programme du ESP01 pour le projet du drone.
 * Serveur TCP qui transmet les commandes de l'application cellulaire au microcontrôleur du drone et qui renvoie à l'application
 * les données de l'accéléromètre qui est sur le drone.
 * Par François Archambault
 * Pour Pierre Bergeron
 * Session Automne 2020
*/
//section include
#include <ESP8266WiFi.h>
#include <WiFiClient.h>


int port = 8888;  //Port number
WiFiServer server(port);//objet serveur de classe WifiServer (pour serveur TCP)

//Server connect to WiFi Network
const char *ssid = "DSO";  //Enter your wifi SSID
const char *password = "247-367-sh";  //Enter your wifi Password

int count=0;
String command; //string qui va être envoyée au client TCP (l'application) et qui va contenir les vleurs de l'accéléromètre
String dataReceive; //variable qui va accueillir les caractères qui arrivent par le port série

void setup()
{
```

```
  Serial.begin(115200); //pour débugger et aussi pour envoyer les informations par le port série (baudrate: 115200)

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password); //Connexion au wifi

  // Wait for connection
  Serial.println("Connecting to Wifi");
  // Attente de connexion de l'ESP8266 au routeur WIFI
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  // Indique l'adresse IP obtenue pour l'ESP8266
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  // Départ du serveur TCP
  server.begin();
  Serial.print("Open a connection to IP:");
  Serial.print(WiFi.localIP()); //affiche l'adresse IP. Utile pour la connexion avec l'apploication cellulaire
  Serial.print(" on port ");
  Serial.println(port); //8888
}
```

```
void loop()
{
  WiFiClient client = server.available(); // Ouverture d'un socket vers le client que se connecte au serveur TCP

  if (client) // Le client existe?
  {
    if(client.connected()) // Le client est bien connecté?
    {
      while(client.connected()) //tant que le client est connecté
      {
        while(client.available()>0)  // Lecture de la requête du client (peut être dans plusieurs trames TCP)
        {
         Serial.write(client.read());// Lecture de la requête du client et l'écrit sur le port série pour le microcontrôleur
        }
        while(Serial.available()) //quand quelque chose est transféré par le port série
        {
          dataReceive = (char)Serial.read(); //caractères 1 par 1
          if(dataReceive == "*")//si le caractère == *
          {
            client.println(command); //envoie la trame (accéléromètre) au client (TCP)
            command = ""; //remise à zéro de command
            break; //sortir du while
          }
          else //si c'Est différent d'unr étoile (*)
          {
            command = command + dataReceive; //accumule les informations de l'accéléromètre
```

```
        }

       }

      }

      client.stop(); // Fermeture du socket TCP
    }
  }

}
```

Annexe 7 (ATmega328PB):

```
/*
 * Programme pour l'ATmega328PB pour le projet du drone. Le programme entend de recevoir une commande par le port série. La commande
 * va être une direction ou bien une vitesse. Le programme contrôle les moteurs et l'accéléromètre. Une fois que la commande a été exécuté, le programme
 * envoie par le port série les valeurs de l'accéléromètre pour qu'elles soient affichées dans l'application cellulaire.
 * Par François Archambault
 * Pour Pierre Bergeron
 * Session Automne 2020
 */
//include
#include<Wire.h>
#include <math.h>
//******Moteur 1**********
#define delVerteM1 4 //choisir le sens
#define delRougeM1 7
#define enableM1 3 //le pwm est appliqué sur cette pin

//******Moteur 2**********
#define delVerteM2 10//choisir le sens
#define delRougeM2 8
#define enableM2 5//le pwm est appliqué sur cette pin

//******Moteur 3**********
#define delVerteM3 A3//choisir le sens
#define delRougeM3 A0
#define enableM3 6//le pwm est appliqué sur cette pin
```

```
//******Moteur 4***********
#define delVerteM4 A2//choisir le sens
#define delRougeM4 A1
#define enableM4 9//le pwm est appliqué sur cette pin

const int MPU=0x68;//pour accéléromètre
int16_t AcX,AcY,AcZ; //variables qui vont accueillir les données de l'accéléromètre
String command; //variable qui va accueillir la commande reçue par le port série
String dataReceive; //va accuillir 1 par 1 les caractères reçus par le port série avant qu'il soit mis dans command
String valeurACC = ""; //variable string qui va accuillir la string qui sera envoyée par le port série pour envoyer les valeurs de l'accéléromètre
String myStringX; //les trois variables pour aider à la conversion en string des valeurs int x, y, z
String myStringY;
String myStringZ;



void setup()
{
  Serial.begin(115200); //port série baudrate de 115200
  Wire.begin(); //pour accéléromètre
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  //définition des moteurs comme des sorties
  pinMode(delVerteM1, OUTPUT);
  pinMode(delRougeM1, OUTPUT);
```

```
  pinMode(enableM1, OUTPUT);

  pinMode(delVerteM2, OUTPUT);
  pinMode(delRougeM2, OUTPUT);
  pinMode(enableM2, OUTPUT);

  pinMode(delVerteM3, OUTPUT);
  pinMode(delRougeM3, OUTPUT);
  pinMode(enableM3, OUTPUT);

  pinMode(delVerteM4, OUTPUT);
  pinMode(delRougeM4, OUTPUT);
  pinMode(enableM4, OUTPUT);
 //va juste allumer les dels vertes donc tous les moteurs vont aller du même sens
  digitalWrite(delVerteM1, HIGH);
  digitalWrite(delVerteM3, HIGH);
  digitalWrite(delRougeM1, LOW);
  digitalWrite(delRougeM3, LOW);
  digitalWrite(delVerteM2, HIGH);
  digitalWrite(delVerteM4, HIGH);
  digitalWrite(delRougeM2, LOW);
  digitalWrite(delRougeM4, LOW);

}
void loop()
{
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
```

```
  Wire.requestFrom(MPU,14,true);
  int AcXoff,AcYoff,AcZoff;
  int myVitesse;
  char arrayVitesse[5];


  while(Serial.available())
  {
    dataReceive = (char)Serial.read();//caractères 1 par 1
    if(dataReceive == "*")//si le caractère == *  Ca veut dire que c'est une direction qui a été envoyée
    {
      break; //sortir du while
    }
    else if(dataReceive == "/")//si le caractère == /  Ca veut dire que c'est une vitesse qui a été envoyée
    {
      break; //sortir du while
    }
    else
    {
      command = command + dataReceive;//accumule les informations de la commande (ex: Front)
    }
  }


    AcXoff = 0; //pour correction du x y z (dans code de base de l'accéléromètre)
    AcYoff = 300;
    AcZoff = 0;
```

```
  //read accel data
  AcX=(Wire.read()<<8|Wire.read()) + AcXoff;
  AcY=(Wire.read()<<8|Wire.read()) + AcYoff;
  AcZ=(Wire.read()<<8|Wire.read()) + AcYoff;

  myStringX = String(AcX); //met les ints en strings pour pouvoir les envoyer par le port série
  myStringY = String(AcY);
  myStringZ = String(AcZ);
  valeurACC = " " + myStringX + " " + myStringY + " " + myStringZ + " " + "crap8266/*" + " ";
//création de la string qui sera envoyée comme réponse à la commande reçue

  if(command.length() <4 && command.length() > 0) //si c'Est une vitesse valide
  {
    command.toCharArray(arrayVitesse,command.length()+1); //string en tableau de char
    myVitesse = (atoi(arrayVitesse)); //tableau de char en int pour pouvoir utiliser la vitese avec
analogWrite

    if(myVitesse == 0) //éteint le drone
    {
     Serial.println(valeurACC);
     analogWrite(enableM3, 0);
     analogWrite(enableM4, 0);
     analogWrite(enableM1, 0);
     analogWrite(enableM2, 0);
    }
    else //pWM avec la vitesse sur les 4 moteurs
    {
     Serial.println(valeurACC);
     analogWrite(enableM3, myVitesse);
```

```
      analogWrite(enableM4, myVitesse);
      analogWrite(enableM1, myVitesse);
      analogWrite(enableM2, myVitesse);
    }
  }

  if(command == "Front") //si la commande reçue est Front
  {
    Serial.println(valeurACC); //envoie les valeurs de l'accéléromètre

    analogWrite(enableM3, (myVitesse + 150));//ajuste la vitesse des moteurs avec la direction voulue
    analogWrite(enableM4, (myVitesse + 150));
    analogWrite(enableM1, myVitesse);
    analogWrite(enableM2, myVitesse);
    command =""; //remet la commande à zéro
  }
  else if(command == "Back")//si la commande reçue est Back
  {
    Serial.println(valeurACC); //envoie les valeurs de l'accéléromètre
    analogWrite(enableM1, (myVitesse + 150)); //ajuste la vitesse des moteurs avec la direction voulue
    analogWrite(enableM2, (myVitesse + 150));
    analogWrite(enableM3, myVitesse);
    analogWrite(enableM4, myVitesse);
    command =""; //remet la commande à zéro
  }
  else if(command == "Gauche")//si la commande reçue est Gauche
  {
```

```
    Serial.println(valeurACC); //envoie les valeurs de l'accéléromètre


    analogWrite(enableM2, (myVitesse + 150));//ajuste la vitesse des moteurs avec la direction
voulue
    analogWrite(enableM4, (myVitesse + 150));
    analogWrite(enableM1, myVitesse);
    analogWrite(enableM3, myVitesse);
    command ="";//remet la commande à zéro
  }
  else if(command == "Droite")//si la commande reçue est Droite
  {
    Serial.println(valeurACC); //envoie les valeurs de l'accéléromètre


    analogWrite(enableM1, (myVitesse + 150));//ajuste la vitesse des moteurs avec la direction
voulue
    analogWrite(enableM3, (myVitesse + 150));
    analogWrite(enableM2, myVitesse);
    analogWrite(enableM4, myVitesse);
    command ="";//remet la commande à zéro
  }


 command ="";//remet la commande à zéro
 valeurACC = ""; //remet les valeurs de l'accéléromètre à zéro pour ne pas renvoyer les mêmes
données
 delay(100);
}
```