Dossier ressources: Testeur de batterie AR.Drone Parrot

1. Découverte du produit et de la problématique technique

L'utilisateur de drone dispose en général de plusieurs batteries lui permettant une utilisation prolongée en remplaçant une batterie déchargée par une chargée, évitant ainsi d'attendre le temps de la charge. Même si l'interface de pilotage indique le niveau de charge de la batterie, il faut un certain temps pour obtenir l'information (installation de la batterie, démarrage du drone et de l'application puis attente de connexion).

Le produit proposé est un testeur de batterie pour AR. Drone Parrot. Ce testeur permet de connaître instantanément le taux de charge d'une batterie exprimé en % et de savoir si cette batterie est suffisamment chargée pour offrir l'autonomie permettant au drone d'effectuer un vol nommé « vol type » défini ci-dessous :

Décollage	5 minutes de vol	Atterrissage
		

Figure 1 : définition d'un « vol type »

Ces informations sont affichées sur un écran LCD RGB.

Les batteries utilisées sont de type Li-Po (Lithium Polymère) de tension nominale $U_{nominale}$ d'environ 12 V.

Pour déterminer le taux de charge d'une batterie, il suffit de mesurer la tension présente à ses bornes. Celle-ci varie légèrement en fonction du taux de charge selon la courbe de décharge donnée en figure 2.

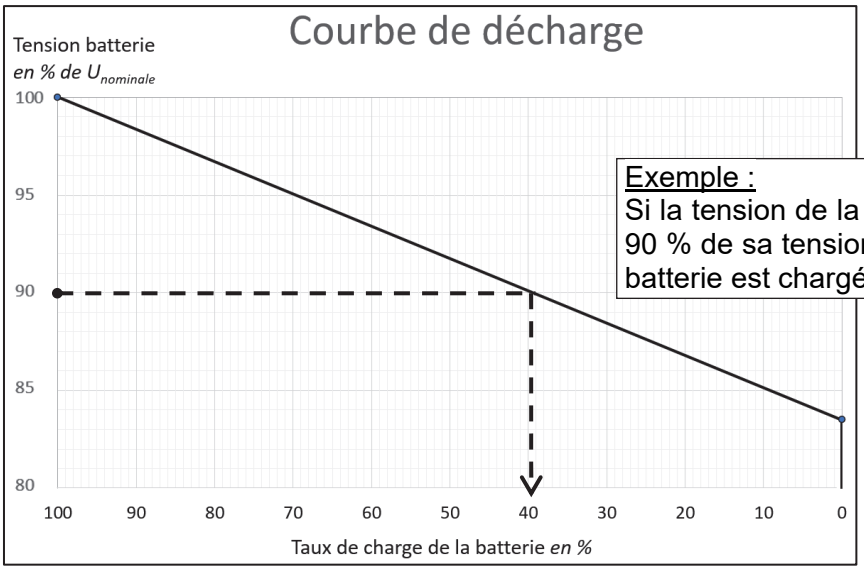


Figure 2 : courbe de décharge

En plus d'afficher le taux de charge de la batterie, un code couleur du rétroéclairage de l'écran LCD permet de savoir instantanément si la batterie est vide, insuffisamment ou suffisamment chargée pour un « vol type ». En cas d'absence de batterie, l'écran affiche un message sur fond bleu.

a. Fonctions du testeur

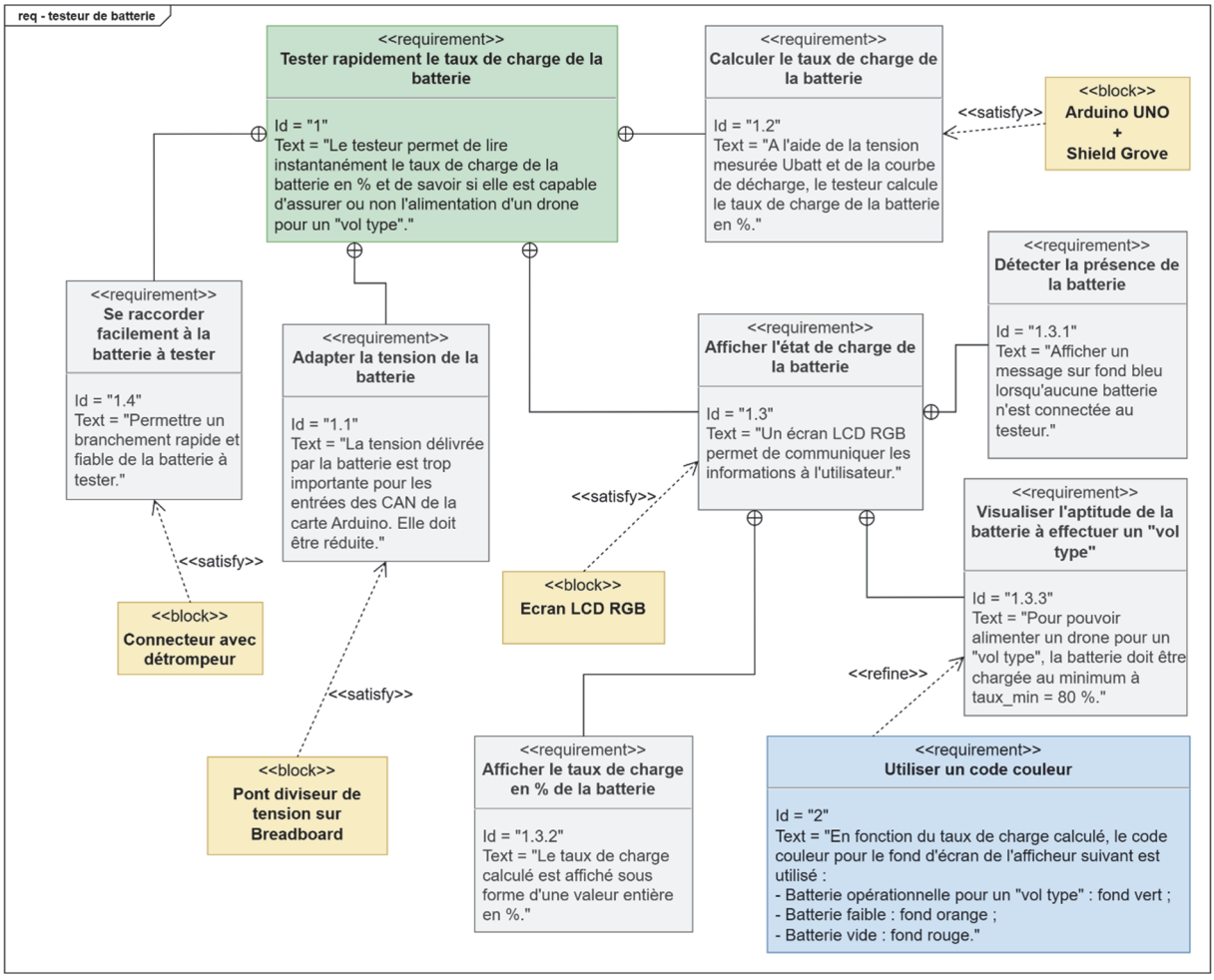


Figure 3 : diagramme d'exigences

b. Structure du testeur

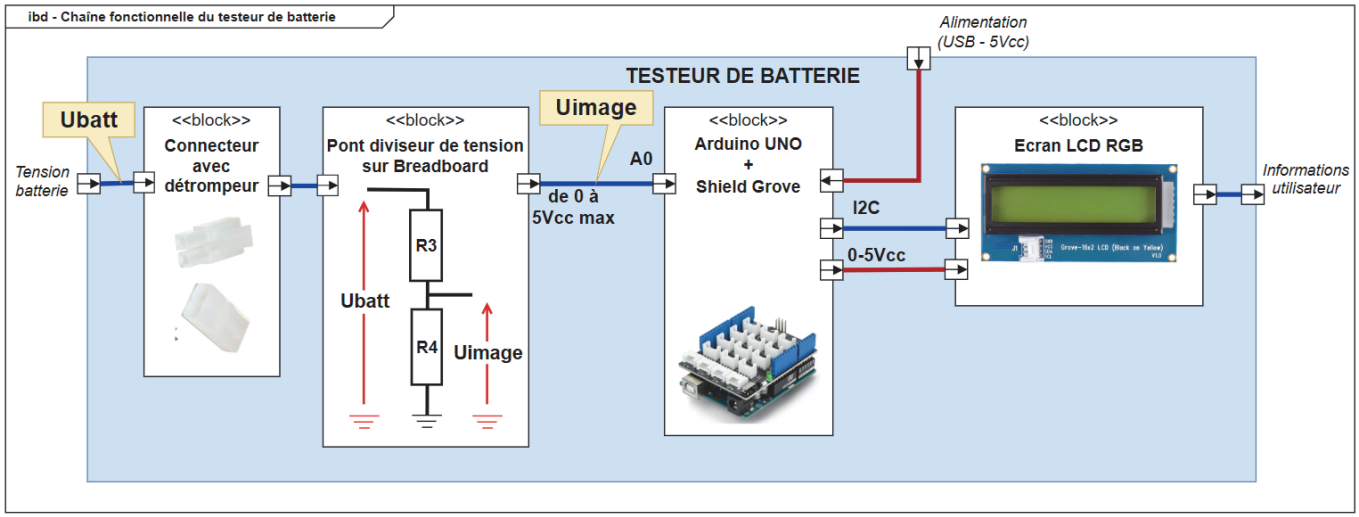
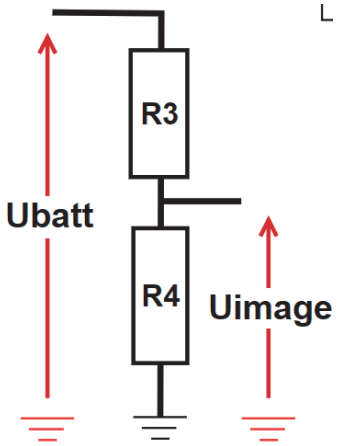


Figure 4 : diagramme de bloc interne

2. Conception

a. Choix du pont diviseur de tension

Paramètres et conditions limites :



Facteur de division :	Pour obtenir une tension U_{image}=5 V en sortie du pont diviseur pour U_{batt}=12 V, il faut un facteur de division k=0,40 ± 5%
Impédance du pont diviseur :	Une impédance R_e< 10 kΩ permet de garantir le bon fonctionnement du CAN de la carte Arduino.
Puissance dissipée :	Une puissance dissipée P < 0,25 W permet de garantir l'intégrité des résistances *R3* et *R4*.

Figure 5 : paramètres du pont diviseur

Configurations disponibles et paramètres associés :

	***R3* [Ω]**	***R4* [Ω]**	**R_e [Ω]**	***k***	***P* [W]**
Configuration n°1	120	47	33,8	0,28	0,77
Configuration n°2	180	120	72	0,4	0,43
Configuration n°3	10 000	10 000	5 000	0,5	0,0065
Configuration n°4	22 000	15 000	8 919	0,405	0,0035
Configuration n°5	68 000	47 000	27 791	0,409	0,0011
Configuration n°6	100 000	150 000	60 000	0,6	0,0005

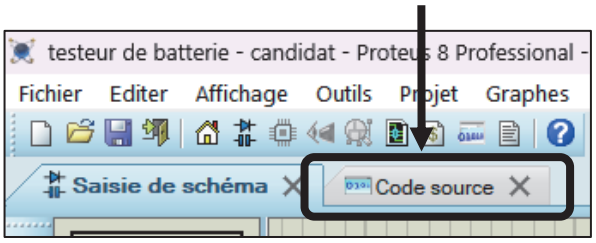
Figure 6 : configurations disponibles

b. Fichier de modélisation du testeur de batterie

La modélisation du testeur de batterie réalisée sous Proteus est disponible dans le dossier ressources dans le fichier :

« Testeur de batterie - candidat.pdsprj »

à enregistrer au nom du candidat



L'onglet « Code source » permet de saisir le programme de pilotage du modèle.

Les zones à compléter sont signalées par une série d'étoiles « ******… » :

Remarque : une zone à compléter peut nécessiter une ou plusieurs instructions.

```
#include <Wire.h>       //bibliothèque I2D
#include "rgb_lcd.h"    //bibliothèque écran LCD RGB

//Définition des constantes
*****************************
const int taux_min = ******************************;
const float Unominale = 12;
const float R3 = ******************************;
```

c. Comportement attendu du testeur

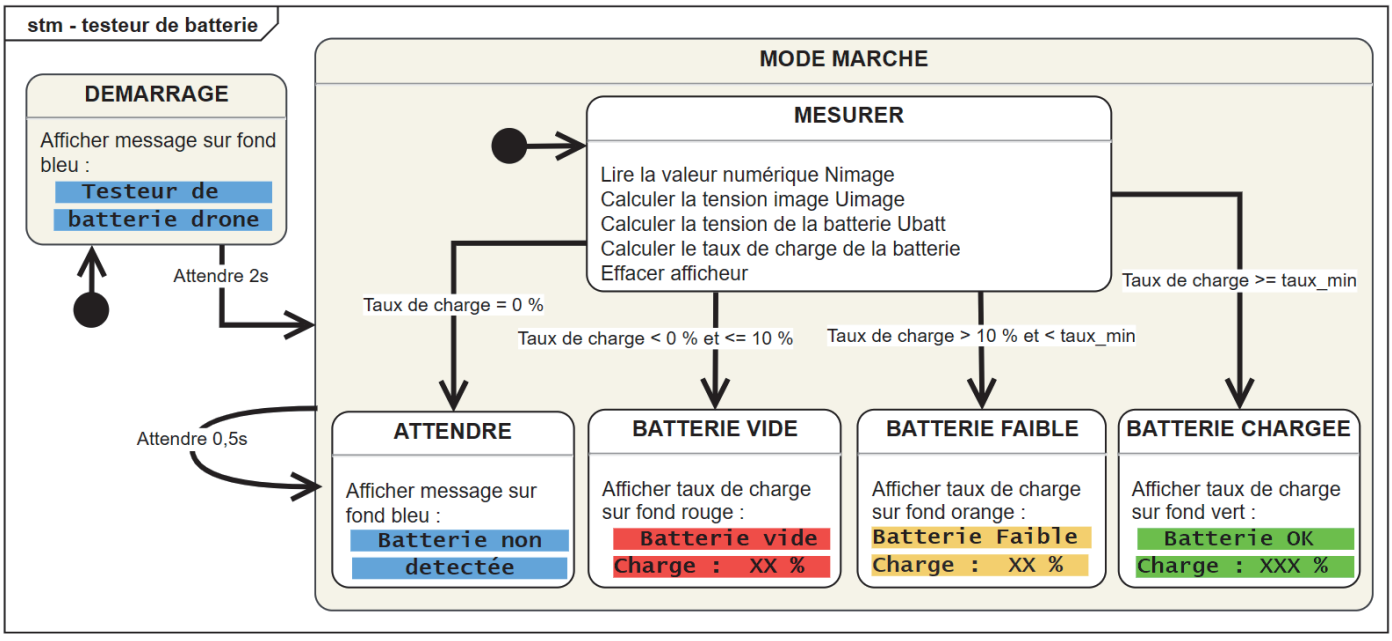


Figure 7 : diagramme d'états

d. Syntaxes de base C++ Arduino

	FONCTION	SYNTAXE
Mode de broche	Le pinMode est une fonction qui permet d'indiquer à la carte Arduino si la broche sera une entrée (INPUT) ou une sortie (OUTPUT).	`pinMode(n°PIN, INPUT)` `pinMode(n°PIN, OUTPUT)`

	SYNTAXE LECTURE	SYNTAXE ECRITURE
Entrée/ sortie digitale	`digitalRead(n°PIN)` Retourne l'état logique 0 ou 1	`digitalWrite(n°PIN, HIGH)` → mise à l'état 1 `digitalWrite(n°PIN, LOW)` → mise à l'état 0
Entrée/ sortie analogique	`analogRead(n°PIN)` Retourne une valeur numérique entre 0 et 1023 proportionnelle à la tension en entrée (entre 0 et 5V).	Commande PWM : 0 ≤ Value ≤ 255 `analogWrite(n°PIN, Value)` `analogWrite(n°PIN, 0)` → tension = 0V `analogWrite(n°PIN, 255)` → tension = 5V

	TEST	SYNTAXE C++ ARDUINO
Tests logiques	Si n est égal à zéro	`if (n == 0){…}`
	Si n supérieur ou égal à 0	`if (n >= 0) {…}`
	Si n est différent de 34	`if (n != 34) {…}`
	Si n est compris strictement entre 0 et 10	`if ((n > 0) && (n < 10)) {…}`
	Si n est inférieur ou égal à 10 ou supérieur ou égal à 50	`if ((n <= 10) || (n >= 50)) {…}`

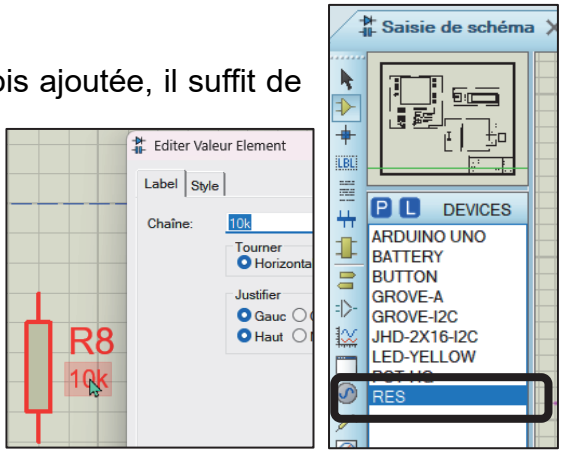
e. Codage des couleurs en hexadécimal

Couleur	Code décimal	Code HEX
ROUGE	255, 0, 0	#FF0000
ORANGE	255, 127, 0	#FF7F00
JAUNE	255, 255, 0	#FFFF00
VERT	0, 255, 0	#00FF00
BLEU	0, 0, 255	#0000FF
VIOLET	143, 0, 255	#8F00FF

Figure 8 : codes couleurs

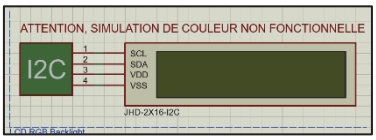
3. Simulation

Sous Proteus, une résistance se nomme RES. Une fois ajoutée, il suffit de double-cliquer pour pouvoir la paramétrer.



Remarque :

Sous Proteus, la simulation de la couleur d'éclairage de l'écran LCD ne fonctionne pas.



Protocole de simulation à suivre :

a) **Ouvrir** le fichier « Tableau de simulation » disponible dans le dossier ressources et **l'enregistrer** au nom du candidat ;

3 Simulation : Tableau de simulation

Cases jaunes = valeurs à saisir
Cases grises = ne pas modifier le calcul automatique

$U_{nominale}$ (en V) = 12

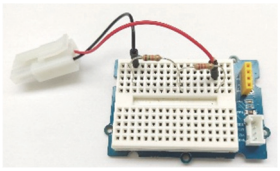
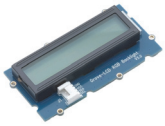
Taux de charge de la batterie en %	Tension de batterie en % de $U_{nominale}$	Tension calculée $U_{nominale}$ en V	Message affiché sur l'écran LCD
100%		0,0	
80%		0,0	
40%		0,0	
6%		0,0	
0%		0,0	

b) Pour chaque taux de charge de la batterie en % :

- **relever** la valeur « Tension de batterie en % de $U_{nominale}$ » à l'aide du graphique « courbe de décharge » (voir Figure 2) ;
- **noter** cette valeur dans le tableau ;
- Sous Proteus, **paramétrer** la tension de batterie avec la valeur « Tension calculée $U_{nominale}$ en V » calculée par le tableau ;
- **lancer** la simulation ;
- **compléter** la colonne « Message affiché sur l'écran LCD » dans le tableau.

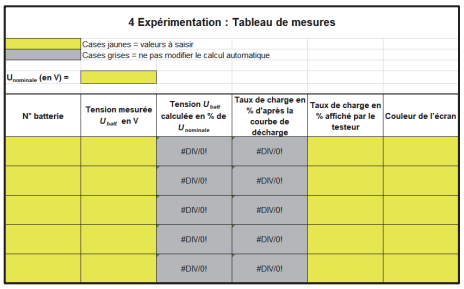
4. Expérimentation

Pour effectuer l'expérimentation, le matériel suivant est à disposition :

 Carte Arduino Uno R3	 Shield base Grove	 Un pont diviseur câblé sur platine	
 3 batteries (dont une chargée à 100%)	 Un multimètre	 Ecran LCD RGB Grove	 Câble Grove et fils de câblage

Protocole d'expérimentation à suivre :

- **Mesurer** les valeurs réelles des résistances *R3* et *R4* du pont diviseur fourni ;
- **Raccorder** le pont diviseur fourni et l'écran LCD à la carte Adruino ;
- **Mesurer** la tension nominale $U_{nominale}$ d'une batterie chargée à 100 % ;
- Après avoir mis sous tension la carte Arduino, **mesurer** la tension de référence 5 V de la carte Arduino ($U_{ref\_arduino}$) ;
- **Ajuster** les valeurs des paramètres $U_{nominale}$, *R3*, *R4*, $U_{ref\_arduino}$ du programme ;
- **Téléverser** le programme dans le testeur de batterie ;
- **Ouvrir** le fichier « Tableau de mesures » disponible dans le dossier ressources et l'**enregistrer** au nom du candidat ;
- **Reporter** la valeur de la tension $U_{nominale}$ dans le tableau ;
- Pour chaque batterie à tester :
  - **reporter** le numéro de la batterie dans le tableau ;
  - **mesurer** la tension de la batterie Ubatt et la **reporter** dans le tableau ;
  - **brancher** la batterie sur le testeur ;
  - **lire** le taux de charge affiché par le testeur et le **reporter** dans le tableau ;
  - **identifier** la couleur du fond d'écran du testeur et la **reporter** dans le tableau.



4 Expérimentation : Tableau de mesures

Cases jaunes = valeurs à saisir
Cases grises = ne pas modifier le calcul automatique

$U_{nominale}$ (en V) =

N° batterie	Tension mesurée U_{batt} en V	Tension U_{batt} calculée en % de $U_{nominale}$	Taux de charge en % d'après la courbe de décharge	Taux de charge en % affiché par le testeur	Couleur de l'écran
		#DIV/0!	#DIV/0!		
		#DIV/0!	#DIV/0!		
		#DIV/0!	#DIV/0!		
		#DIV/0!	#DIV/0!		
		#DIV/0!	#DIV/0!		