



Espruino

(Partie 1 : prise en main)

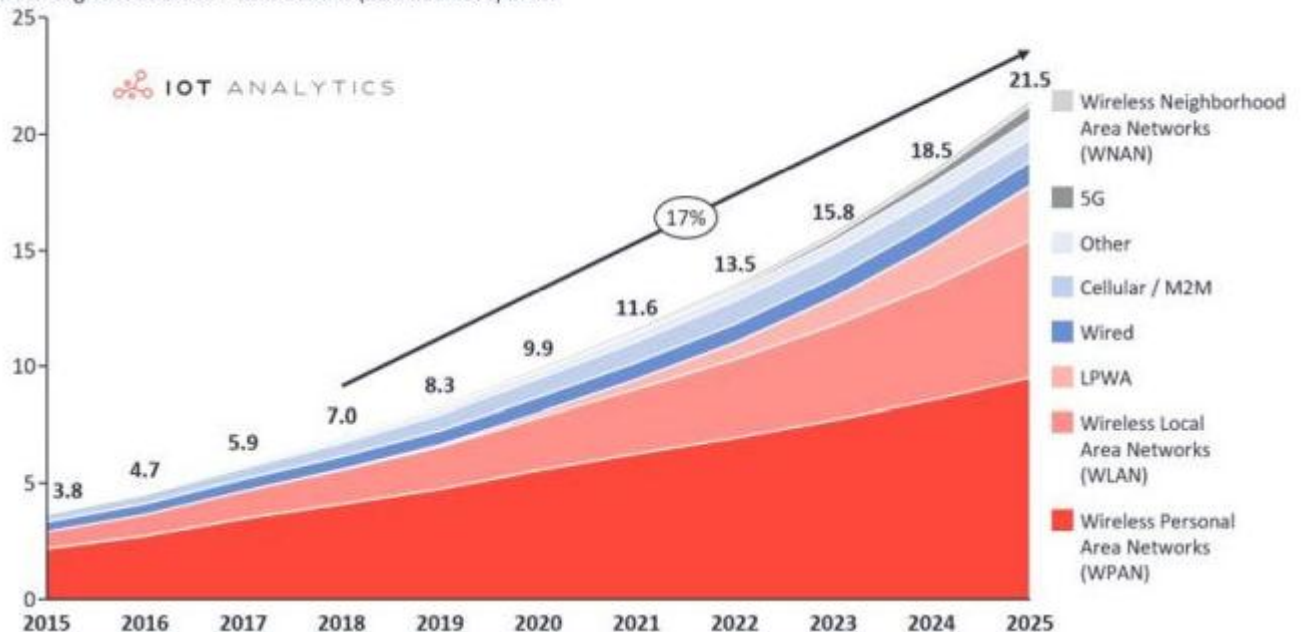


LE CREN Anthony

Bientôt tous les appareils que vous possédez, et pratiquement tous les objets qui existent seront connectés à l'Internet. Que ce soit via votre téléphone portable, des vêtements ou des appareils ménagers, nous serons connectés à l'Internet des objets (IoT). Le nombre d'objets actifs connectés à un réseau IoT était de 7 milliards d'unités dans le monde en 2018 et passera à 21,5 milliards en 2025. Cela est considéré comme la troisième évolution de l'Internet, baptisé Web 3.0 qui fait suite à l'ère du Web social.

Global Number of Connected IoT Devices

Number of global active IoT Connections (installed base) in Bn

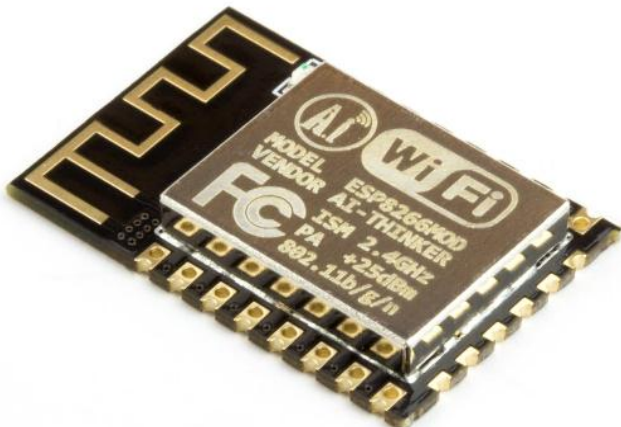


Rappel :

- Le Web 1.0 est le Web constitué de pages web liées entre elles par des hyperliens et qui a été créé au début des années 1990.
- Le Web 2.0 est le Web social, qui s'est généralisé avec le phénomène des blogs, réseaux sociaux et la technologie wiki (Un wiki est une application web qui permet la création, la modification et l'illustration collaboratives de pages à l'intérieur d'un site web).

LPWAN : Low-power wide-area network (Liaisons sans fil à faible consommation énergétique)	WIFI (Wireless Fidelity) dans un WLAN
<ul style="list-style-type: none"> - Fréquence : 868Mhz - Puissance 20mW - Portée 100km max - Débit : 100 bit/s 	<ul style="list-style-type: none"> - Fréquence : 2.4Ghz - Puissance 100mW - 802.11g : 54Mbps/s, 140m max - 802.11n : 450Mbps/s, 250m max


Dans ce nouveau monde des objets connectés, le module ESP8266 est devenu une solution populaire et abordable. Cette carte électronique est un SOC (System On Chip, Système sur une puce) qui intègre un processeur Xtensa LX106, une interface RF (Radio Fréquence) et une pile TCP/IP qui permet d'implémenter un support WiFi.

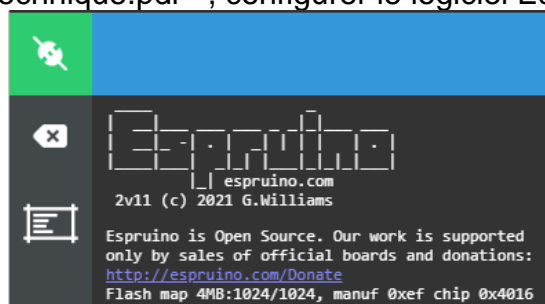
Caractéristiques de l'ESP8266 (NodeMCU, ESP-12E)		
Fréquence	80Mhz	
Flash Memory	4 Mbytes	
RAM	80 Kbytes	
Broches E/S	11	
Entrée Analogique	1	
WIFI/PWM/SPI/I2C	oui	
Alimentation	3.3V	

Description des périphériques montés sur la carte Shield

Numéro de broche (OUT/IN)		Composant	Rôle
0	Logique (OUT)	Led rouge	Led de test
2	Logique (OUT)	Sortie RS485	Commande de spots DMX
4	Logique	SDA : Afficheur OLED	Afficheur OLED et capteur I2C divers
5	Logique	SCL : Afficheur OLED	
12	Logique	DS18S20	Capteur de température
13	Logique (IN)	VS1838B	Capteur infrarouge pour télécommande
14	Logique (OUT)	WS2812B	Ruban de 4 leds couleurs
15	Logique (OUT)	Emetteur UHF	Passerelle vers le réseau Sigfox
A0	Analogique (IN)	Capteur LDR	Capteur de lumière

Rem : Un Shield est une carte qui se branche sans soudure aux cartes ESP-12E pour augmenter leurs capacités.

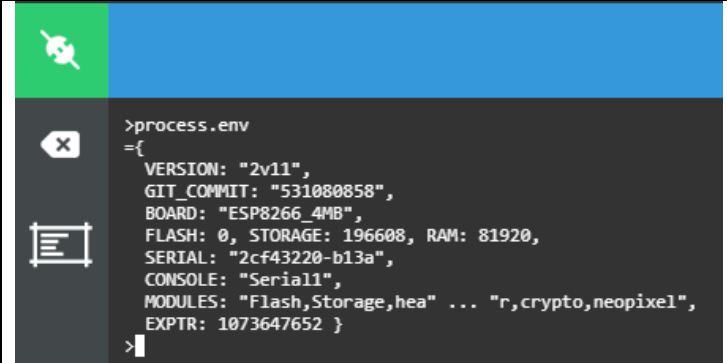
A d'aide du fichier « Espruino technique.pdf », configurer le logiciel Espruino web IDE 



1 Quelques commandes dans la console

`process.env`

http://www.espruino.com/Reference#t | process_env

 <pre>>process.env ={ VERSION: "2v11", GIT_COMMIT: "531080858", BOARD: "ESP8266_4MB", FLASH: 0, STORAGE: 196608, RAM: 81920, SERIAL: "2cf43220-b13a", CONSOLE: "Serial1", MODULES: "Flash,Storage,hea" ... "r,crypto,neopixel", EXPTR: 1073647652 } ></pre>	<p>Affiche la version du firmware et de la référence du processeur utilisé</p>
---	--

`process.memory()`

http://www.espruino.com/Reference#t | process_memory

<pre>>process.memory() ={ free: 1571, usage: 29, total: 1600, history: 18, gc: 0, gctime: 2.805, blocksize: 12 } ></pre>	<p>Affiche la configuration mémoire ne lien avec le processeur utilisé.</p>
--	---

`process.env.MODULES.split(',')`

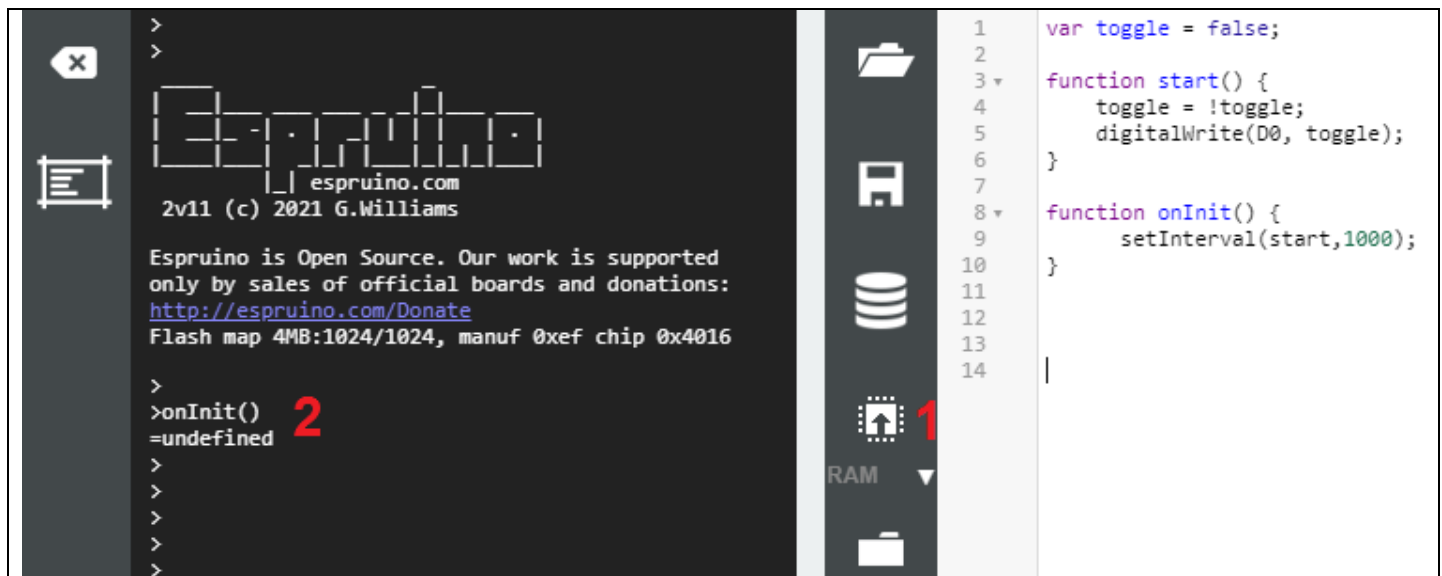
<pre>>process.env.MODULES.split(',') =["Flash", "Storage", "heatshrink", "net", "dgram", "http", "NetworkJS", "Wifi", "ESP8266", "TelnetServer", "crypto", "neopixel"] ></pre>	<p>Affiche les modules (bibliothèques) gérés en natif avec le firmware</p>
--	--

Les modules sont appelés avec la fonction `require`, ex : `require("ESP8266").getState()`

<http://www.espruino.com/Reference#ESP8266>

1 Clignoter une led

Tester le programme suivant : led.js



1 : charger le programme dans l'esp8266

2 : exécuter le programme en saisissant dans la console `onInit()`

```

>onInit()
=undefined

```

Astuce :

Utiliser l'auto-complétion dans la console, c'est-à-dire taper les deux caractères « on », puis la touche de tabulation du clavier. Il ne restera plus qu'à ajouter les deux parenthèses.

Remarque :

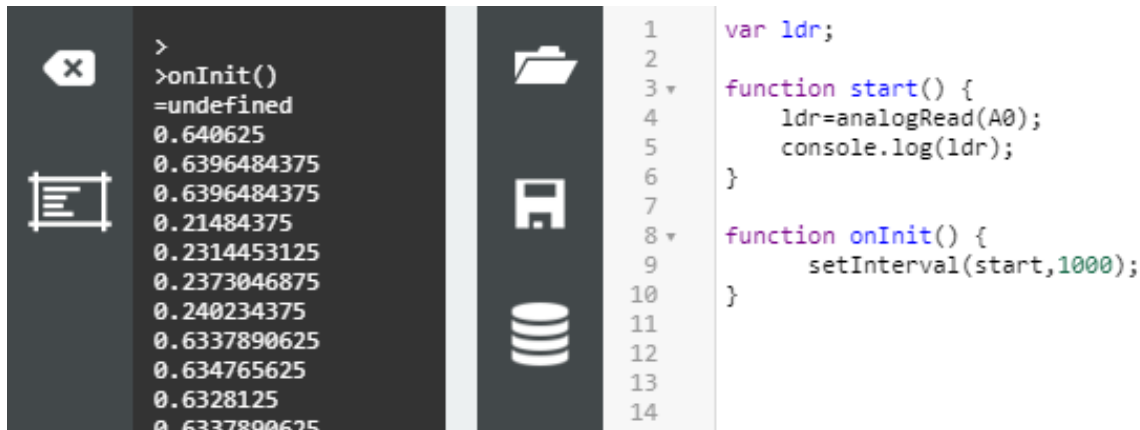
L'affichage de `undefined` est normal puisque la fonction `onInit()` ne retourne rien.

Q1 Que faut-il modifier pour allumer et éteindre la led toutes les 500 millisecondes ? Vérifier expérimentalement votre solution.

Changer le paramètre 1000 par 500

2 Mesurer la luminosité

Tester le programme suivant : ldr.js



```

1  var ldr;
2
3  function start() {
4      ldr=analogRead(A0);
5      console.log(ldr);
6  }
7
8  function onInit() {
9      setInterval(start,1000);
10 }
11
12
13
14

```

Console output:

```

>
>onInit()
=undefined
0.640625
0.6396484375
0.6396484375
0.21484375
0.2314453125
0.2373046875
0.240234375
0.6337890625
0.634765625
0.6328125
0.6337890625

```

Q2 Compléter le tableau suivant

Valeur moyenne relevée avec la main au-dessous du capteur (Masque la lumière)	Valeur moyenne relevée avec un éclairage nominal (Avec de la lumière)
0.21	0.72

Q3 Calculer la moyenne des deux valeurs précédentes

$$(0.21+0.72)/2=0.46$$

Q4 On désire commander la led avec le capteur de lumière de la manière suivante :

- Présence de la lumière sur le capteur : led allumée
- Absence de lumière sur le capteur : led éteinte

Réaliser un programme (Q4-lumiere_led.js) en s'inspirant des deux programmes précédents

```

var ldr;

function start() {
    ldr=analogRead(A0);
    console.log(ldr);
    if (ldr>0.5){
        digitalWrite(D0, true);
    }
    else
    {
        digitalWrite(D0, false);
    }
}

function onInit() {
    setInterval(start,1000);
}

```

2 Ruban de 4 leds couleurs

Tester le programme suivant : neopixels.js

```

1  var np = new Uint8ClampedArray(4*3);
2  var npPin = 14;
3
4  function setLed(n,r,v,b){
5      np[n*3]=v;
6      np[n*3+1]=r;
7      np[n*3+2]=b;
8  }
9
10 function clearLed() {
11     for (var i=0;i<np.length;i++) {
12         np[i]=0;
13     }
14     require("neopixel").write(npPin, np);
15 }
16
17 function start() {
18     clearLed();
19     setLed(0,0, 16, 0);
20     setLed(1,0, 32, 32);
21     setLed(2,16, 0, 16);
22     setLed(3,16, 16, 16);
23     require("neopixel").write(npPin, np);
24 }
  
```

Q5 A quoi peuvent correspondre les paramètres de la fonction setLed dans le programme ci-dessus ?

n,r,v,b : numéro de la led , composante rouge, composante verte, composante bleue

Q6 Sachant que la composante r,v,b d'une led peut aller de 0 à 255, quel est le nombre de couleurs affichables sur une led ?

$256 \times 256 \times 256 = 16777216$ couleurs

Q7 Modifier le programme précédent conformément au tableau ci-dessous : (Q7-neopixels_coul.js)

LED0	LED1	LED2	LED3
ROUGE	VERT	BLEU	JAUNE

```

setLed(0,16, 0, 0);
setLed(1,0, 16, 0);
setLed(2,0, 0, 16);
setLed(3,16, 16, 0);
  
```

Q8 On désire allumer toutes les leds de la même couleur. Ecrire un programme utilisant une boucle Pour (Q8-neopixels_une_coul.js)

```
function start() {
  clearLed();
  var i;
  for (i=0;i<4;i++){
    setLed(i,16, 16, 0);
  }
  require("neopixel").write(npPin, np);
}
```

Q9 Compléter la fonction défilement qui permet de faire défiler les leds avec une couleur verte comme un chenillard. (L'intervalle de temps sera de 100ms entre chaque led)

LED0	LED1	LED2	LED3

(Q9-neopixels_defill.js)

```
var np = new Uint8ClampedArray(4*3);
var npPin = 14;
var etape=0;

function setLed(n,r,v,b){
  np[n*3]=v;
  np[n*3+1]=r;
  np[n*3+2]=b;
}

function clearLed() {
  for (var i=0;i<np.length;i++) {
    np[i]=0;
  }
  require("neopixel").write(npPin, np);
}

function start() {
  var i;
  for (i=0;i<4;i++){
    if (i==etape){
      setLed(i,16, 16, 0);
    }
    else{
      setLed(i,0, 0, 0);
    }
  }
  etape=(etape+1)%4;
  require("neopixel").write(npPin, np);
}

function onInit() {
  clearLed();
  setInterval(start,100);
}
```

Q10 On désire commander les leds neopixels avec le capteur de lumière de manière progressive

- Présence de la lumière sur le capteur : led rouge vif (16)
- Absence de lumière sur le capteur : led rouge sombre (proche de zéro)

Réaliser un programme (Q10-lumiere_neopixels.js) en s'inspirant des programmes Q2 et Q8

Rappel :

	min	Max
Capteur de lumière	0	1
Valeur d'une composante	0	255

La valeur d'une composante couleur ne doit pas dépasser 16 pour cet exercice

```
var np = new Uint8ClampedArray(4*3);
var npPin = 14;

function setLed(n,r,v,b) {
  np[n*3]=v;
  np[n*3+1]=r;
  np[n*3+2]=b;
}

function clearLed() {
  for (var i=0;i<np.length;i++) {
    np[i]=0;
  }
  require("neopixel").write(npPin, np);
}

function start() {
  var i,ldr;
  ldr=analogRead(A0);
  for (i=0;i<4;i++){
    setLed(i,ldr*16, 0, 0);
  }
  require("neopixel").write(npPin, np);
}

function onInit() {
  clearLed();
  setInterval(start,10);
}
```


3 Afficheur OLED

La définition de l'afficheur est de 128x64 pixels

Tester le programme suivant : Q11-oled.js

```
I2C1.setup({ 'scl': 5,
             'sda': 4,
             bitrate: 100000 });
var g = require("SSD1306").connect(I2C1);

function start() {
  g.setFontVector(20);
  g.drawString("Hello NSI",0,0);
  // write to the screen
  g.flip();
}
```

Q11 Déplacer le texte affiché approximativement au centre de l'écran. Où sont les informations de position x et y dans la méthode text ?

```
g.drawString("Hello NSI",15,20);
```

Q12 Réaliser un défilement du texte du haut vers le bas. (Q12-oled_defill.js)

```
I2C1.setup({ 'scl': 5,
             'sda': 4,
             bitrate: 100000 });
var g = require("SSD1306").connect(I2C1);
var y=0;

function start() {
  y++;
  if (y>50){
    y=0;
  }
  g.clear();
  g.drawString("Hello NSI",15,y);
  // write to the screen
  g.flip();
}

function onInit() {
  g.setFontVector(20);
  setInterval(start,1);
}
```

Q13 Réaliser un défilement du texte du haut vers le bas puis du bas vers le haut sans coupure (Q13-
oled_defill_haut_bas.js)

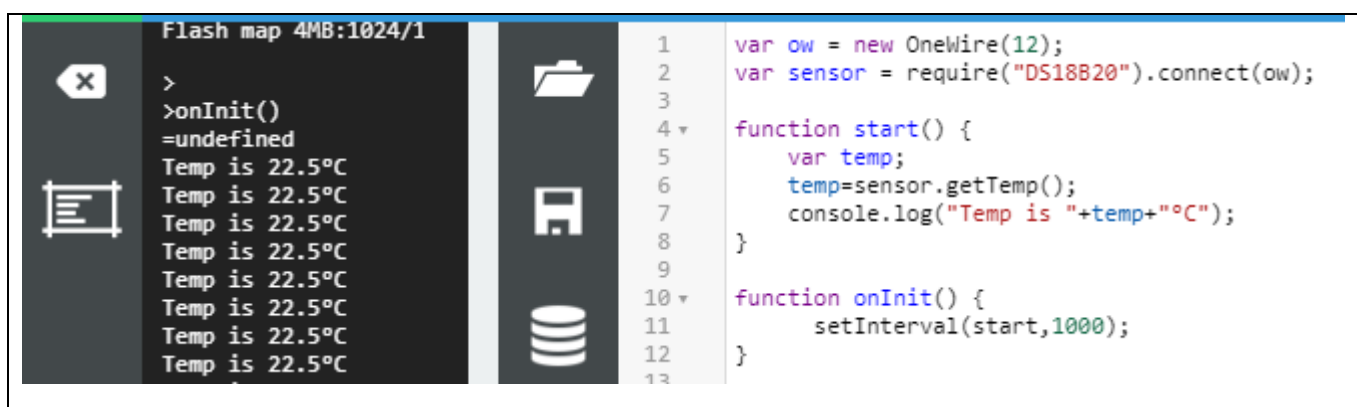
```
I2C1.setup({ 'scl': 5,
             'sda': 4,
             bitrate: 100000 });
var g = require("SSD1306").connect(I2C1);
var y=0;
var sens=0;

function start() {
  if (sens==0){
    y+=2;
    if (y>50){
      sens=1;
    }
  }
  else{
    y-=2;
    if (y<0){
      sens=0;
    }
  }
  g.clear();
  g.drawString("Hello NSI",15,y);
  // write to the screen
  g.flip();
}

function onInit() {
  g.setFontVector(20);
  setInterval(start,1);
}
```

4 Mesure de température

Tester le programme suivant : temperature.js



```
Flash map 4MB:1024/1
>
>onInit()
=undefined
Temp is 22.5°C
Temp is 22.5°C
Temp is 22.5°C
Temp is 22.5°C
Temp is 22.5°C
Temp is 22.5°C
Temp is 22.5°C
Temp is 22.5°C
Temp is 22.5°C
Temp is 22.5°C

1  var ow = new OneWire(12);
2  var sensor = require("DS18B20").connect(ow);
3
4  function start() {
5    var temp;
6    temp=sensor.getTemp();
7    console.log("Temp is "+temp+"°C");
8  }
9
10 function onInit() {
11   setInterval(start,1000);
12 }
13
```

Q14 Afficher la température sur l'écran OLED (Q14-oled_temperature.js)

La méthode `text` de l'afficheur ne gère que les chaînes de caractères. Pour afficher un nombre, utiliser la méthode `toString()`.

Exemple :

```
a=10 ;
g.drawString(a.toString(),0,0);
```

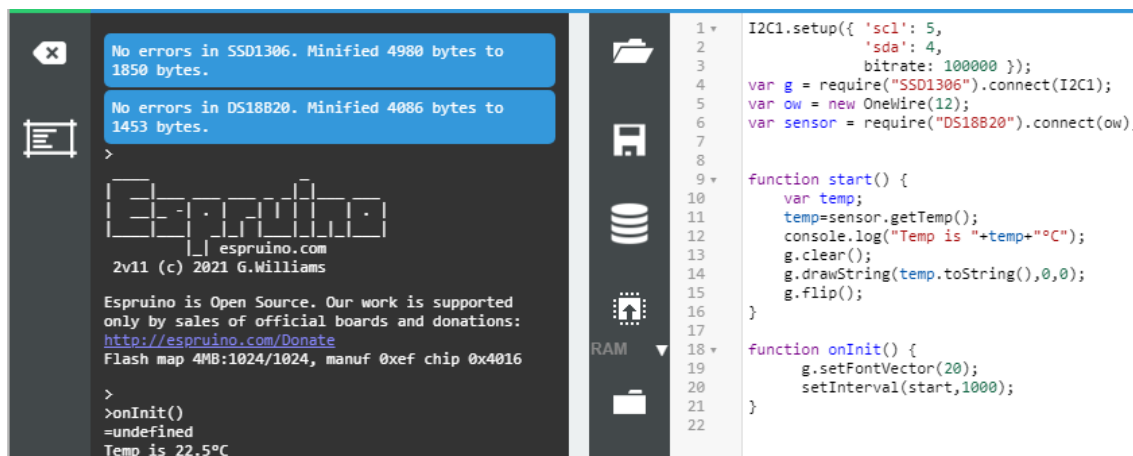
```
I2C1.setup({ 'scl': 5,
             'sda': 4,
             bitrate: 100000 });
var g = require("SSD1306").connect(I2C1);
var ow = new OneWire(12);
var sensor = require("DS18B20").connect(ow);

function start() {
  var temp;
  temp=sensor.getTemp();
  console.log("Temp is "+temp+"°C");
  g.clear();
  g.drawString(temp.toString(),0,0);
  g.flip();
}

function onInit() {
  g.setFontVector(20);
  setInterval(start,1000);
}
```

Remarque :

Avant de charger le programme, effectuer la commande `reset()` dans la console afin d'éviter les erreurs de chargement.



Mini projets

Q15 Défiler des couleurs différentes sur le ruban de led

Q16 Déclencher le défilement du ruban de leds en fonction de la luminosité ambiante

Q17 Les couleurs du ruban changent en fonction de la température ambiante

Q18 Gérer l'afficheur OLED en ajoutant :

- La température ambiante
- La température maximale
- La température minimale

Q19 Clignoter la led rouge plus ou moins rapidement en fonction de la luminosité ambiante

Q20 Réaliser un dégradé de couleurs avec le ruban de leds

Q21 Réaliser un compteur binaire (0 à 15) avec le ruban de leds

Q22 Tracer un cercle sur l'afficheur OLED et afficher l'information de température au centre du cercle.

Q23 Votre projet personnel (le programme de votre choix)

Annexe

Sauvegarder le programme en mémoire

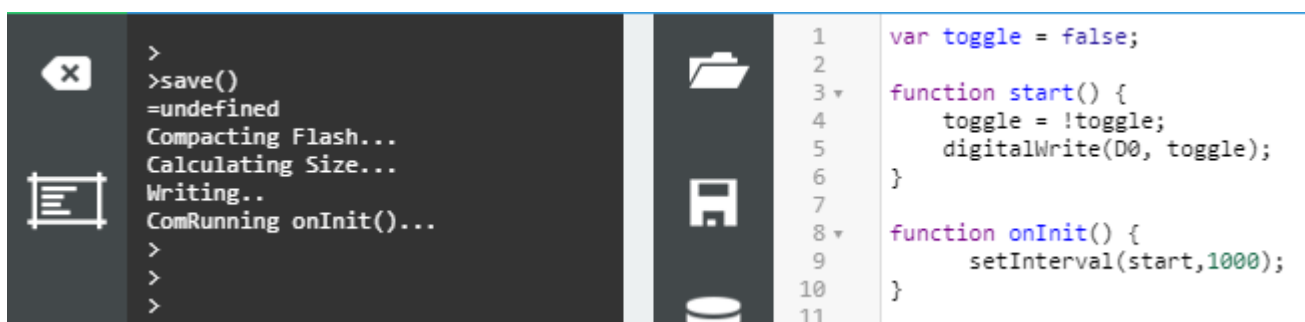
Un programme peut être sauvegardé en mémoire flash et être exécuté automatiquement à la mise sous tension. Pour cela on utilise la bibliothèque « Storage ». Il est absolument nécessaire que le programme possède une fonction `onInit()`.

<https://www.espruino.com/Reference#Storage>

Commencer par « formater » l'espace dédié aux fichiers utilisateur

```
>require('Storage').eraseAll();
=undefined
>require("Storage").getFree();
=196608 //bytes free
```

Charger le programme led.js



<pre>>dump() var toggle = false; function start() { toggle = !toggle; digitalWrite(D0, toggle); } function onInit() {setInterval(start,1000);} // Code saved with E.setBootCode var toggle = false; function start() { toggle = !toggle; digitalWrite(D0, toggle); } function onInit() { setInterval(start,1000); } =undefined ></pre>	<p>Vérifier la présence du programme en mémoire flash avec la commande <code>dump()</code></p>
--	--

<pre>>save() =undefined Compacting Flash... Calculating Size... Writing.. ComRunning onInit()... ></pre>	<p>Dans la console, utiliser la fonction <code>save()</code> pour sauvegarder le code en mémoire flash.</p> <p>Le programme démarre automatiquement, la led clignote.</p>
--	---

Redémarrer le programme sauvegardé sans débrancher la carte :

<pre>>E.reboot() =undefined ets Jan 8 2013,rst cause:2, boot mode:(3,6) load 0x40100000, len 2408, room 16 tail 8 chksum 0xe5 load 0x3ffe8000, len 776, room 0 tail 8 chksum 0x84 load 0x3ffe8310, len 632, room 0 tail 8 chksum 0xd8 csum 0xd8 2nd boot version : 1.6 SPI Speed : 40MHz SPI Mode : DIO SPI Flash Size & Map: 32Mbit(1024KB+1024KB) jump to run user1 @ 1000 (Eâini:sÂoo iddld'b> ÿ<("løgäfoäd' (Eäs'dÄdpLoading 1150 bytes from flash Running onInit()... > </pre>		<p>E.reboot()</p> <p>Le programme n'est pas effacé</p> <p>La led doit clignoter de nouveau.</p>
--	--	---

Pour effacer le code, utiliser la fonction **reset(true)**

<pre>>reset(true) Erasing saved code. Done! =undefined Espruino _ espruino.com 2v11 (c) 2021 G.Williams Espruino is Open Source. Our work is supported only by sales of official boards and donations: http://espruino.com/Donate Flash map 4MB:1024/1024, manuf 0xef chip 0x4016 > </pre>	<pre>>reset(true) Erasing saved code. Done! =undefined Espruino _ espruino.com 2v11 (c) 2021 G.Williams Espruino is Open Source. Our work is supported only by sales of official boards and donations: http://espruino.com/Donate Flash map 4MB:1024/1024, manuf 0xef chip 0x4016 ></pre>
--	---

Il est possible de sauvegarder en mémoire flash le programme sans passer par la RAM. Pour cela il faut sélectionner la destination avant le transfert.

