

- Microcontrôleur
- Langage C++
- Bibliothèques
  - WIFI, client web
  - WIFIAP, WebServeur
  - HTTP, REST
- Application

Les microcontrôleurs sont fréquemment utilisés dans les systèmes embarqués : télécommandes, jouets, la téléphonie mobile. . .

Les systèmes embarqués :

- ont des contraintes de taille (intégration), de consommation électrique (autonomie) de coût (grande série)
- sont en général affectés à une tâche bien précise et ont une taille des programmes et de mémoire (vive et morte) réduite
- communiquent avec des dispositifs d'entrées-sorties (IO) : boutons, relais, résistances variables, optocoupleurs, moteurs électriques, LED, circuits intégrés logiques. . .
- n'ont parfois aucun dispositif d'interface homme-machine : ni clavier, ni écran, ni disque.

## Un microcontrôleur intègre :

- un processeur (CPU)
- de la mémoire vive (RAM) pour stocker les données et variables
- de la mémoire morte (ROM) pour stocker le programme.
- souvent un oscillateur pour le cadencement (quartz ou LC = circuit self-condensateur)
- des périphériques, capables d'effectuer des tâches spécifiques

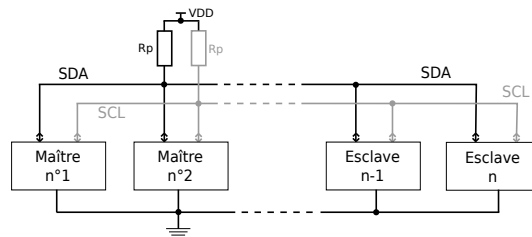
## Exemple de périphériques :

- Convertisseurs Analogiques-Numériques (CAN) (donnent un nombre binaire à partir d'une tension électrique)
- Convertisseurs Numériques-Analogiques (CNA) (effectuent l'opération inverse)
- les générateurs de signaux à Modulation de Largeur d'Impulsion (MLI) (pour faire du son par exemple)
- des timers/compteurs
- des chiens de garde (watchdog)
- des comparateurs (de tensions électriques)
- des contrôleurs de bus de communication (UART, I2C, SSP, CAN, FlexRay, USB...).

I2C (Inter-Integrated Circuit) est une communication avec 3 fils :

- SDA (Serial Data Line) : ligne de données bidirectionnelle
- SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle
- La masse

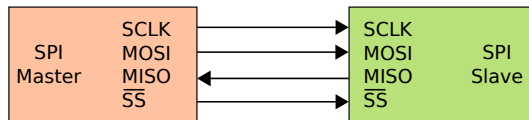
Le pavé tactile (touchpad) du WT32-SCO1 utilise le protocole I2C.



SPI (Serial Peripheral Interface) de la famille des SSP (Synchronous Serial Peripheral) est une communication avec 5 fils :

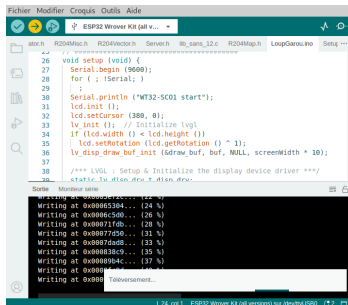
- SCLK — Serial Clock, Horloge (produit par le maître)
- MOSI — Master Output, Slave Input (produit par le maître)
- MISO — Master Input, Slave Output (produit par l'esclave)
- SS — Slave Select, Actif à l'état bas (produit par le maître)
- La masse

L'écran à cristaux liquides (LCD screen) du WT32-SCO1 utilise le protocole SPI.



Le port USB est utilisé comme un port série et possède 2 fonctions principales

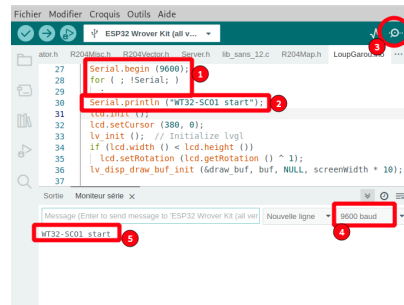
- La programmation
- La communication



The screenshot shows the Arduino IDE with the 'ESP32 Wrover Kit (all versions)' selected. The code in the main window includes a `void setup()` function that initializes the serial port at 9600 baud and prints 'WT32-SC01 start'. The serial monitor at the bottom shows the output: 'WT32-SC01 start'.

```
26 void setup() {  
27   Serial.begin(9600);  
28   for (; !Serial;)  
29     ;  
30   Serial.println("WT32-SC01 start");  
31   lcd.init();  
32   lcd.setCursor(380, 0);  
33   lv_init(); // Initialize lvgl  
34   if (lcd.width() < lcd.height())  
35     lcd.setRotation(lcd.getRotation() ^ 1);  
36   lv_disp_draw_buf_init(&draw_buf, buf, NULL, screenWidth * 10);  
37  
38   /** LVGL : Setup & Initialize the display device driver **/  
39   static lv_disp_drv_t disp_drv;
```

Sortie Moniteur série  
Writing at 0x00053804... (24 %)  
Writing at 0x0006c500... (26 %)  
Writing at 0x00071f0b... (28 %)  
Writing at 0x00077d50... (31 %)  
Writing at 0x0007dad8... (33 %)  
Writing at 0x000838c9... (35 %)  
Writing at 0x00089b4c... (37 %)  
Writing at 0x0009...  
Writing at 0x000... Téléversement...



The screenshot shows the same Arduino IDE interface as the previous one, but with red boxes and numbers highlighting specific elements: 1. `Serial.begin(9600);`, 2. `Serial.println("WT32-SC01 start");`, 3. The serial monitor icon in the top right, 4. The '9600 baud' dropdown menu, and 5. The 'WT32-SC01 start' output in the serial monitor.

```
27   Serial.begin(9600);  
28   for (; !Serial;)  
29     ;  
30   Serial.println("WT32-SC01 start");  
31   lcd.init();  
32   lcd.setCursor(380, 0);  
33   lv_init(); // Initialize lvgl  
34   if (lcd.width() < lcd.height())  
35     lcd.setRotation(lcd.getRotation() ^ 1);  
36   lv_disp_draw_buf_init(&draw_buf, buf, NULL, screenWidth * 10);  
37
```

Sortie Moniteur série x  
Message (Enter to send message to 'ESP32 Wrover Kit (all ver...)  
Nouvelle ligne 9600 baud  
WT32-SC01 start

Il n'y a pas de système d'exploitation

- Programme unique
- Pas de préemption
- Ne jamais écrire d'étreinte fatale (deadlock)

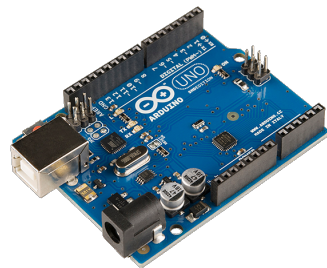
Il y a deux phases

- Initialisation ("setup")
- Boucle générale ("loop")
  - Affichage
  - Saisie
  - Réseau

L'Arduino emprunte son nom au “Bar di Re Arduino” (“bar du roi Arduino”), lieu de réunion des concepteurs de la carte en Italie du Nord.

Il y a une centaine d'instruction qui s'exécutent en générale en 1 cycle d'orloge, soit 1 Mips par MHz.

Un module Arduino est généralement construit autour d'un microcontrôleur Atmel AVR (de type Harvard, c'est à dire avec séparation des bus de programme et de données) qui fonction à une tension de 5 V.

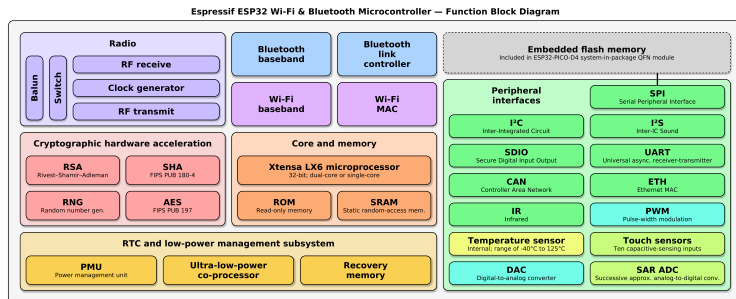




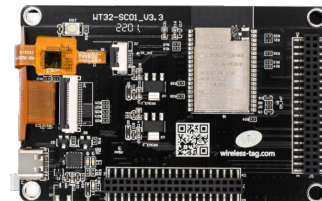
L'espressif 32 n'est pas un Arduino. C'est un microcontrôleurs intégrant la gestion du Wi-Fi, du Bluetooth et un DSP (processeur de signal numérique).

Le support du développement peut se faire avec l'IDE d'Arduino (par extension).

Il est adapté au domaine de l'Internet des objets.



- Module sans fil – ESP32-WROVER-B module s'appuyant sur processeur ESP32-D0WD dual-core à 240 Mhz avec mémoire flash 4MB SPI, 8MB PSRAM, 2.4GHz WiFi 802.11 b/g/n à 150 Mbps, et Bluetooth 4.2 LE (faible consommation)
- Affichage – 3.5 pouce 480×320 (SPI) avec pavé tactile capacitif 2-broches (I2C)
- Expansion – 2x 40 broches I/O avec GPIO, I2C, VN/VP, I2S, UART, 5 V, 3.3V, GND
- Divers – Reset button (6), power and UART LEDs
- Alimentation – 5V/1A via port USB type-C (affichage seul), ou 5V/2A lorsque des extension sont connectés
- Dimensions – 90.87 x 58.01 mm



Les mots clefs ne peuvent être utilisés comme identifiant (variable, fonction, namespace)

29 C (K&amp;R)

4 ANSI-C

15 C++

asm	catch	continue	double	float	if	new	public	signed	switch	try	virtual
auto	char	default	else	for	inline	operator	register	sizeof	template	typedef	void
break	class	delete	enum	friend	int	private	return	static	this	union	volatile
case	const	do	extern	goto	long	protected	short	struct	throw	unsigned	while

## Déclaration

<code>extern type v;</code>	prédéclaration de variable
<code>extern type v (val...);</code>	prédéclaration de variable
<code>type f ();</code>	fonction
<code>type f (val...)</code>	fonction

## Définition

<code>[static auto register]</code>	localisation de mémoire
<code>type v;</code>	variable !!!
<code>type v (val...);</code>	variable
<code>type v ();</code>	fonction !!!
<code>type v (type...);</code>	fonction

## Opérateurs

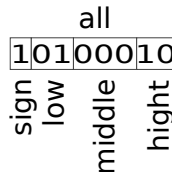
opérateur	fonction	utilisation	asso
::	accès global	:: nom_global	
::	résolution de portée	nom_classe :: nom_membre	g → d
()	parenthésage	(expr)	
[]	indice de tableau	expr [expr]	g → d
()	appel de méthode ou fonction	expr (list_expr <sub>opt</sub> )	g → d
()	construction	nom_type (list_expr)	
.	accès de membre	objet . nom_membre	g → d
->	accès de membre par pointeur	ptr_objet -> nom_membre	g → d
++	incrémement post-f xée	var ++	g → d
--	décrémement post-f xée	var --	g → d
++	incrément pré-f xée	++ var	g ← d
--	décrément pré-f xée	-- var	g ← d
*	déréférencer un pointeur	* pointeur	g ← d
&	adresse d'un objet	& objet	g ← d
+	plus unaire	+ expr	g ← d
-	moins unaire	- expr	g ← d
!	négation	! expr	g ← d
~	complément bit à bit	~ expr	g ← d
sizeof ()	taille des exemplaires d'un type	sizeof ( type )	
sizeof	taille d'un objet	sizeof expr	
new	allocation mémoire et construction	new (list_arg) <sub>opt</sub> type [nb_objets] <sub>opt</sub>	
delete	destruction et libération mémoire	delete [] <sub>opt</sub> pointeur	
()	conversion de type ("cast")	(type) expr	g ← d
.*	accès indirect de membre	objet .* var	g → d
->*	accès indirect de membre par pointeur	ptr_objet -> * var	g → d
*	multiplication	expr * expr	g → d
/	division	expr / expr	g → d

opérateur	fonction	utilisation	asso
%	modulo	expr % expr	g → d
+	addition	expr + expr	g → d
-	soustraction	expr - expr	g → d
<<	décalage de bits à gauche	scalaire << nb_bits	g → d
>>	décalage de bits à droite	scalaire >> nb_bits	g → d
<	inférieur à	expr < expr	g → d
>	supérieur à	expr > expr	g → d
<=	inférieur ou égal à	expr <= expr	g → d
>=	supérieur ou égal à	expr >= expr	g → d
==	égalité	expr == expr	g → d
!=	inégalité	expr != expr	g → d
&	ET bit à bit	expr & expr	g → d
^	OU exclusif bit à bit	expr ^ expr	g → d
	OU inclusif bit à bit	expr   expr	g → d
&&	ET logique	expr && expr	g → d
	OU logique inclusif	expr    expr	g → d
?:	expression conditionnelle	expr ? expr : expr	g ← d
=	affectation	var = expr	g ← d
*=	multiplication et affectation	var *= expr	g ← d
/=	division et affectation	var /= expr	g ← d
%=	modulo et affectation	var %= expr	g ← d
+=	addition et affectation	var += expr	g ← d
-=	soustraction et affectation	var -= expr	g ← d
<<=	décalage à gauche et affectation	var <<= expr	g ← d
>>=	décalage à droite et affectation	var >>= expr	g ← d
&=	ET bit à bit et affectation	var &= expr	g ← d
^=	OU inclusif bit à bit et affectation	var ^= expr	g ← d
=	OU exclusif bit à bit et affectation	var  = expr	g ← d
,	enchaînement	expr , expr	g → d

Priorité des principaux opérateurs (fort → faible)	associativité
()	g → d
[] -> .	g → d
&(adresse) *(pointeur) ±(unaire) ++ -- ! ~ (casting) sizeof	<b>g ← d</b>
*(multiplication) / %	g → d
±(binaire)	g → d
<< >>	g → d
< <= > >= == !=	g → d
& (bit à bit)	g → d
^ (bit à bit)	g → d
(bit à bit)	g → d
&& (logique)	g → d
(logique)	g → d
?:	<b>g ← d</b>
= op= (op ∈ { * / % + - << >> & ^   })	<b>g ← d</b>
,	g → d

La taille des types est connue,... ... mais ils peuvent être définis au bit près

<code>char</code>	1 octet
<code>short</code>	2 octets
<code>int</code>	2 ou 4 octets
<code>long</code>	4 octets
<code>long long</code>	8 octets
<code>float</code>	4 octets
<code>double</code>	8 octets



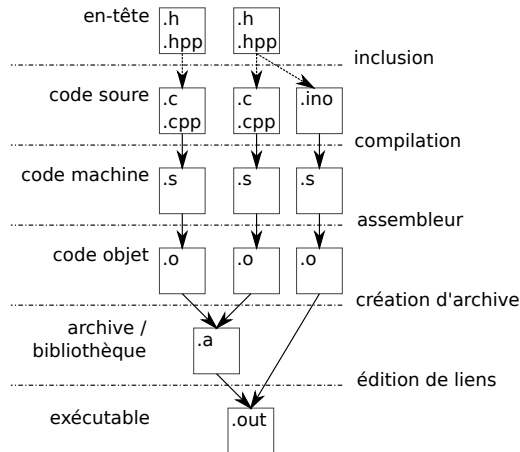
C'est indispensable pour écrire de pilotes qui vont affecter des valeurs sur des lignes d'entrées/sorties.

```

1  union {
2      char all;
3      struct {
4          int sign :1;
5          int hight :2;
6          int middle:3;
7          int low :2;
8      } split;
9  } x;
10
11 x.all = 0xA2;
12 x.split.sign = 1;
13 x.split.hight = 1;
14 x.split.middle = 0;
15 x.split.low = 2;
```

Le compilateur par défaut enchaîne toutes les étapes.

Des options de compilation permettent de conserver les fichiers intermédiaires.





- Pour simplifier le paramétrage de la chaîne de compilation, l'environnement de développement intégré (IDE) compile et fait l'édition de liens de tous les source se trouvant dans un répertoire (c, c++, python...).
- Il faut donc bien séparer les sources des programmes pour des cibles différentes.

```
1 |-- Test.ino
2 |-- font_lib_12.c
3 |-- R204Vector.h
4 |-- Server.h
5 |-- Server.h
```

## Les tableaux

- Un tableau se définit avec les crochets : []
- Cela permet de réserver la mémoire et fournir un pointeur sur le premier élément.
- Ce pointeur ne peut être modifié.
- Il n'y a aucun contrôle sur les index. Une valeur négative ou au-delà de la zone réservée n'est pas garantie (dans le meilleur des cas un plantage).
- En cas de plantage, le matériel recommence à zéro.

```
1  int ti [3] = {1, 2, 3};
2  int j;
3
4  ti [-1]; // =? j
5
6  ti [10]; // ?
7
8  int *pi = ti;
9  int *pi2 = &ti [0];
10
11 ti [2] == pi [2]; // true
12
13 ++pi;
14 ti [2] == *(pi+1); // true
15
16 ++ti; // error
```

Les chaînes de caractères en C sont des tableaux d'octets.

- un octet à zéro détermine la fin.
- leur taille ne peut pas changer.
- mais on peut modifier le contenu.

```
1 void f() {  
2     char tc [] = "bonjour"; // OK init  
3  
4     char *pc = tc;  
5  
6     char *pc2 = "bonjour"; // error  
7     const char *pc3 = "bonjour";  
8 }
```

Allocation dynamique

- new : rend un pointeur
- delete : supprime un pointeur
- delete[] : supprime un tableau

```
1 int *pi = new int;  
2 *pi = 3;  
3  
4 delete pi;  
5  
6 int *pti = new int [3];  
7  
8 delete pti; // ?  
9  
10 delete [] pti;
```

Les vecteurs sont des tableaux dynamiques

- contient les longueurs utile et réservée

ne varifie pas les dépassements

- “at” vérifie les dépassements

Les “string” sont des chaînes dynamiques

- contient la longueur de la chaîne
- fourni des fonctions de traitement de chaîne

```
1 #include <vector>
2 using namespace std;
3 void f() {
4     vector<int> v1 {1, 2, 3 }; // {1, 2, 3}
5     vector<int> v2 (3, 6); // {6, 6, 6}
6     vector<int> v3 (4); // {0, 0, 0, 0}
7
8     vector<int> v4 (v1); // {1, 2, 3}
9 }
```

```
1 string s0 ("Initial string"); // "Initial string"
2 string s1; // ""
3 string s2 (s0); // "Initial string"
4 string s3 (s0, 8, 3); // "str"
5 string s4 (10, 'x'); // "xxxxxxxxxx"
```

Les tables associatives permette

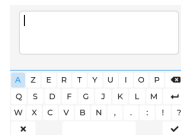
- d'utiliser des index qui ne sont pas numériques.
- d'utiliser un nombre limité de valeurs  
ne vérifie les dépassements
- "at" vérifie les dépassements

```
1 #include <string>
2 #include <map>
3 using namespace std;
4 int main () {
5     map<string, int> map1 = {
6         { "alpha", 0 },
7         { "beta", 1 },
8         { "gamma", 2 }
9     };
10
11     map1 ["alpha"] = 10;
12     map1.at("beta") = 20;
13     map1 ["omega"]; // ?
14
15     map<int, int> mymap = {
16         { 0, 0 },
17         { 100, 2 },
18         { 10000, 4 }
19     };
20
21     return 0;
22 }
```

```

1 #include <LovyanGFX.hpp> // main library
2 static LGFX lcd;        // declare display variable
3 #include <lvgl.h>
4 #include "lv_conf.h"
5 // ...
6
7 void setup (void) {
8   lcd.init (); // Initialize LovyanGFX
9   lv_init (); // Initialize lvgl
10  // ...
11 }
12
13 void loop () {
14   lv_timer_handler (); // let the GUI do its work
15   // ...
16 }

```



```

1 displayFormNet (bool show) {
2   if (!show) {
3     if (formNet)
4       lv_obj_del (formNet);
5     formNet = nullptr;
6     return;
7   }
8   formNet = lv_obj_create (lv_scr_act ());
9   // ...
10  lv_obj_t *listMode (lv_obj_create (formNet));
11  lv_obj_add_event_cb (listMode, callBackSetMode
12    ,
13    LV_EVENT_CLICKED, &selectedMode);
14  // ...
15  lv_obj_set_size (formNet, 350, 200);
16  lv_obj_center (formNet);
17 }

```

Les bibliothèques permettent de profiter de fonctions courantes et de les partager avec plusieurs applications. Si vous saisissez “lvgl arduino”

Si vous saisissez “lvgl arduino” vous trouverez probablement la version 7.11, car elle a été la plus vue. Mais elle n'est pas compatible avec la dernière version <https://docs.lvgl.io/master/>

## Exemple de création

## Gère la connexion à un réseau WIFI

- Connexion au SSID
- Création de socket
- Lecture d'un octet (ou -1)
- Écriture d'un nombre ou une chaîne de caractères

```
1 #include <WiFi.h>
2 #include <WiFiClient.h>
3 // ...
4
5 void setup (void) {
6     WiFi.begin ("my-ssid", "min8char");
7     // ...
8 }
9
10 void loop () {
11     WiFiClient client;
12     if (client.connect (webHost, webPort)) {
13         return;
14         client.println ("message");
15         while (client.connected ()) {
16             if (client.available ())
17                 char c = client.read ();
18             //...
19         }
20         client.stop ();
21     }
22     //...
23 }
```

## Gère un point d'accès WIFI

- Création du SSID
- Création d'un serveur
- Configuration d'un serveur

```
1 #include <WiFi.h>
2 #include <WebServer.h>
3 #include <WiFiClient.h>
4
5 static WebServer webServer (80);
6 // ...
7
8 void setup (void) {
9   WiFi.softAP ("my-ssid", "min8char");
10   webServer.on ("/path1", handlePath1);
11   webServer.onNotFound (handleNotFound);
12   webServer.begin ();
13   // ...
14 }
15
16 void handlePath1 (void) {
17   String arg = webServer.arg ("x");
18   webServer.send (200, "text/html", "<h1>OK</h1>");
19
20   WiFiClient client (webServer.client ());
21   client.flush ();
22   client.stop ();
23 }
24
25 void loop () {
26   webServer.handleClient (); // let the server do its work
27   //...
28 }
```



## Le protocole HTTP

- “méthode”
- URL, URI, QueryString
- entête MIME
- données
- code de retour

### Requête

```
1 GET /index.html?x=1&y=2 HTTP/1.1
2 Host: m4102.parlenet.org
3 Accept-Language: argoTerrien
4 User-Agent: Zorglub
5
6 XXX
```

### Réponse

```
1 HTTP/1.1 200 OK
2 Date: Sun, 5 Oct 2003 11:00:07
3 Server: Apache/2.0.40
4 Accept-Ranges: bytes
5 Content-Length: 2898
6 Connection: close
7 Content-Type: text/html
8
9 <!DOCTYPE HTML PUBLIC...
```

## Application

