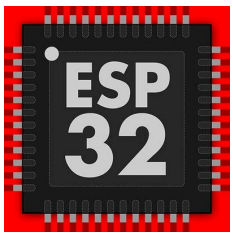
CHARLES CARNUS
LYCÉE TECHNIQUE
ENSEIGNEMENT SUPÉRIEUR



ESP32

TP dans l'IDE Arduino

Prof : Kamal Boudjelaba

14 août 2023

B T S S N

LTP Charles Carnus
Avenue de Saint Pierre - Bourran - 12000 RODEZ, France

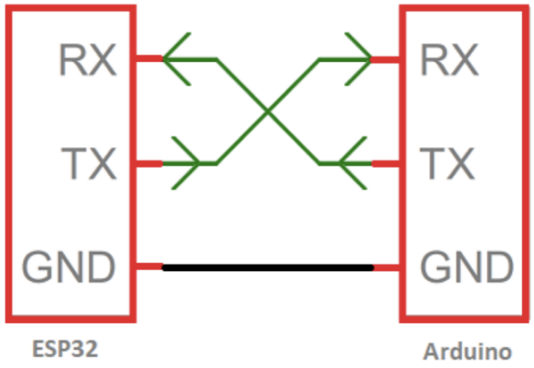
IR–EC

GPIO	INPUT	OUTPUT	Commentaires
0	OUI (Pullup interne)	OUI	Doit être à 0V pendant le FLASH
1 (TX0)	NON	OUI	Communication UART avec le PC
2	OUI (Pulldown interne)	OUI	Doit être à 0V pendant le FLASH
3 (RX0)	OUI	NON	Communication UART avec le PC
4 (D4)	OUI	OUI	
5 (D5)	OUI	OUI	
6	NON	NON	Connecté au flash SPI intégré
7	NON	NON	Connecté au flash SPI intégré
8	NON	NON	Connecté au flash SPI intégré
9	NON	NON	Connecté au flash SPI intégré
10	NON	NON	Connecté au flash SPI intégré
11	NON	NON	Connecté au flash SPI intégré
12 (MTDI) (D12)	OUI (Pulldown interne)	OUI	Doit être à 0V pendant le BOOT
13 (D13)	OUI	OUI	
14 (D14)	OUI	OUI	
15 (MTDO) (D15)	OUI (Pullup interne)	OUI	Startup log si à 3.3V
16 (RX2)	OUI	OUI	Pas dispo sur les WROVER
17 (TX2)	OUI	OUI	Pas dispo sur les WROVER
18 (D18)	OUI	OUI	
19 (D19)	OUI	OUI	
21 (D21)	OUI	OUI	
22 (D22)	OUI	OUI	
23 (D23)	OUI	OUI	
25 (D25)	OUI	OUI	
26 (D26)	OUI	OUI	
27 (D27)	OUI	OUI	
32 (D32)	OUI	OUI	
33 (D33)	OUI	OUI	
34 (D34)	OUI	NON	Pas de pullup/pulldown interne
35 (D35)	OUI	NON	Pas de pullup/pulldown interne
36 (VP)	OUI	NON	Pas de pullup/pulldown interne
39 (VN)	OUI	NON	Pas de pullup/pulldown interne
EN	NON	NON	Relié au bouton EN (ESP32 Reset)

Table 2. I/O possibles pour les pins de l'ESP32

3. TP1 : Communication UART entre Arduino et ESP32

Dans ce TP, on utilise la communication UART entre les cartes de développement Arduino UNO et ESP32.



On utilise Arduino `D1` comme broche `TX` et `D0` comme `RX`. Dans l'ESP32, `GPIO16` comme broche `RX`, et `GPIO17` comme broche `TX`, considérées comme des broches `RX1` et `TX1` dans la carte ESP32.

Code Arduino pour la communication UART BTS SN : KB

```
1
2 void setup() {
3   Serial.begin(9600);
4 }
5
6 void loop() {
7   Serial.println("Charles Carnus");
8   delay(1500);
9 }
```

Code ESP32 pour la communication UART BTS SN : KB

```
1
2 #define RXp2 16
3 #define TXp2 17
4
5 void setup() {
6   Serial.begin(115200);
7   Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);
8 }
9
10 void loop() {
11   Serial.println("Le message reçu est : ");
12   Serial.println(Serial2.readString());
13 }
```

Afficher et décoder les trames envoyées et reçues à l'aide de l'oscilloscope ou de l'analyseur logique.

4. TP2 : Communication UART entre deux ESP32

On utilise la bibliothèque HardwareSerial lorsqu'on travaille avec la communication UART dans l'ESP32 à l'aide des ports UART1 ou UART2.

#include <HardwareSerial.h>

Si on utilise `Serial.begin();` les broches 1 et 3 sont utilisées, ce qui signifie qu'on utilise UART0. Si on veut utiliser d'autres ports série, on doit les définir dans le code. On utilise les commandes suivantes pour définir le port UART sélectionné : Ici, on crée un objet de la bibliothèque HardwareSerial appelé "SerialPort()" et on spécifie le port UART entre parenthèses.

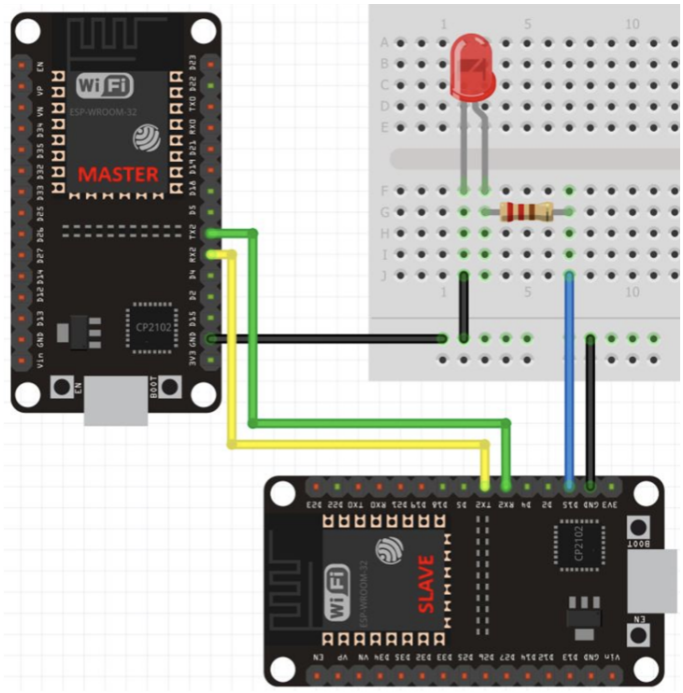
```

Serial.begin() ;                    //si UART0
HardwareSerial SerialPort(1);       //si UART1
HardwareSerial SerialPort(2);       //si UART2
```

Ensuite, dans la fonction `setup()` , on va initialiser la communication UART à l'aide de l'objet `SerialPort` sur la méthode `begin()` et y spécifier quatre paramètres comme suit :
SerialPort.begin(BaudRate, SerialMode, RX_pin, TX_pin);
Les paramètres incluent le débit en bauds pour la communication série, le mode série, la broche RX et la broche TX utilisées.
Dans notre cas, on utilise UART2, et on l'initialisera comme suit :

```

#include <HardwareSerial.h>

HardwareSerial SerialPort(2); // UART2

void setup()
{
  SerialPort.begin(15200, SERIAL_8N1, 16, 17);
}
```

4.1 Manipulation

La communication série (UART) où le maître ESP32 enverra soit "1" soit "0" à l'esclave ESP32. L'esclave recevra alors ces données et contrôlera une LED connectée à sa broche numérique. On utilisera UART2 (comme mentionné précédemment) pour communiquer entre les deux cartes.



On connecte la broche TX2 de la carte maître ESP32 à la broche RX2 de la carte esclave ESP32. De même, on connecte la broche RX2 de la carte maître ESP32 à la broche TX2 de la carte esclave ESP32. La broche de l'anode de la LED est connectée à la broche numérique 15 (esclave) via une résistance de limitation de courant de 220 Ω. La broche de la cathode est mise à la terre. On connecte également le GND des deux cartes ESP32.

Sketch Arduino ESP32 Maître BTS SN : KB

```
1
2 #include <HardwareSerial.h>
3
4 HardwareSerial SerialPort(2); // UART2
5
6 void setup()
7 {
8   SerialPort.begin(15200, SERIAL_8N1, 16, 17);
9 }
10
11 void loop()
12 {
13   SerialPort.print(1);
14   delay(5000);
15   SerialPort.print(0);
16   delay(5000);
17 }
```

Sketch Arduino ESP32 Esclave BTS SN : KB

```
1
2 #include <HardwareSerial.h>
3
4 HardwareSerial SerialPort(2); // UART2
5
6 char number  = ' ';
7 int LED = 15;
8
9 void setup()
10 {
11   SerialPort.begin(15200, SERIAL_8N1, 16, 17);
12   pinMode(LED, OUTPUT);
13 }
14 void loop()
15 {
16   if (SerialPort.available())
17   {
18     char number = SerialPort.read();
19     if (number == '0') {
20       digitalWrite(LED, LOW);
21     }
22     if (number == '1') {
23       digitalWrite(LED, HIGH);
24     }
25   }
26 }
```

La LED s'allumera et restera allumée pendant 5 secondes avant de s'éteindre pendant 5 secondes. Cela continue car le maître continue d'envoyer 1/0 à l'esclave. Afficher et décoder les trames envoyées et reçues à l'aide de l'oscilloscope ou de l'analyseur logique.