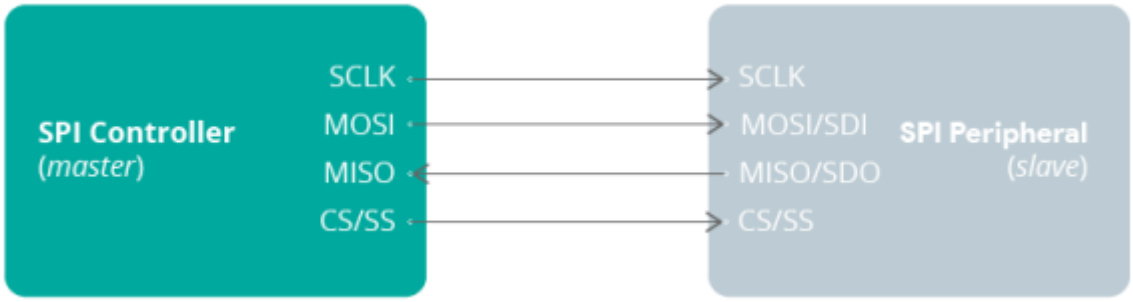
Bus SPI sur ESP32

Présentation du protocole de communication ESP32 SPI

SPI signifie S erial P eripheral Interface, et c'est un protocole de données série synchrone utilisé par les microcontrôleurs pour communiquer avec un ou plusieurs périphériques. Par exemple, votre carte ESP32 communiquant avec un capteur supportant le SPI ou avec un autre microcontrôleur.

Dans une communication SPI, il y a toujours un contrôleur (également appelé maître) qui contrôle les périphériques (également appelés esclaves). Les données peuvent être envoyées et reçues simultanément. Cela signifie que le maître peut envoyer des données à un esclave et qu'un esclave peut envoyer des données au maître en même temps.



Vous ne pouvez avoir qu'un seul maître , qui sera un microcontrôleur (l'ESP32), mais vous pouvez avoir plusieurs esclaves. Un esclave peut être un capteur, un écran, une carte microSD, etc., ou un autre microcontrôleur. Cela signifie que vous pouvez avoir un ESP32 connecté à plusieurs capteurs, mais le même capteur ne peut pas être connecté à plusieurs cartes ESP32 simultanément.

Interface SPI

Pour la communication SPI, vous avez besoin de quatre lignes :

- **MISO** : Master In Slave Out
- **MOSI** : sortie maître entrée esclave
- **SCK** : Horloge Série
- **CS / SS** : Chip Select (utilisé pour sélectionner l'appareil lorsque plusieurs périphériques sont utilisés sur le même bus SPI)

Sur un appareil uniquement esclave, comme les capteurs, les écrans et autres, vous pouvez trouver une terminologie différente :

- MISO peut être étiqueté comme SDO (Serial Data Out)
- MOSI peut être étiqueté comme SDI (Serial Data In)

Périphériques ESP32 SPI

L'ESP32 intègre 4 périphériques SPI : SPI0, SPI1, SPI2 (communément appelé HSPI) et SPI3 (communément appelé VSPI).

SP0 et SP1 sont utilisés en interne pour communiquer avec la mémoire flash intégrée, et vous ne devez pas les utiliser pour d'autres tâches.

Vous pouvez utiliser HSPI et VSPI pour communiquer avec d'autres appareils. HSPI et VSPI ont des signaux de bus indépendants et **chaque bus peut piloter jusqu'à trois esclaves SPI.**

Broches SPI par défaut ESP32

De nombreuses cartes ESP32 sont livrées avec des broches SPI par défaut pré-assignées. Le mappage des broches pour la plupart des cartes est le suivant :

SPI	MOSI	MISO	SCLK	CS
VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

Attention : selon la carte que vous utilisez, les broches SPI par défaut peuvent être différentes. Assurez-vous donc de vérifier le brochage de la carte que vous utilisez. De plus, certaines cartes n'ont pas de broches SPI pré-assignées, vous devez donc les définir sur le code.

Remarque : généralement, lorsqu'elles ne sont pas spécifiées, la carte utilise les broches VSPI lors de l'initialisation d'une communication SPI avec les paramètres par défaut.

Trouver les broches SPI par défaut de votre carte ESP32

Si vous n'êtes pas sûr des broches SPI par défaut de votre carte, vous pouvez télécharger le code suivant pour le savoir.

brochesspi.ino

```
/*
  Rui Santos
  Complete project details at
https://RandomNerdTutorials.com/esp32-spi-communication-arduino/

  Permission is hereby granted, free of charge, to any person obtaining
a copy
  of this software and associated documentation files.
```

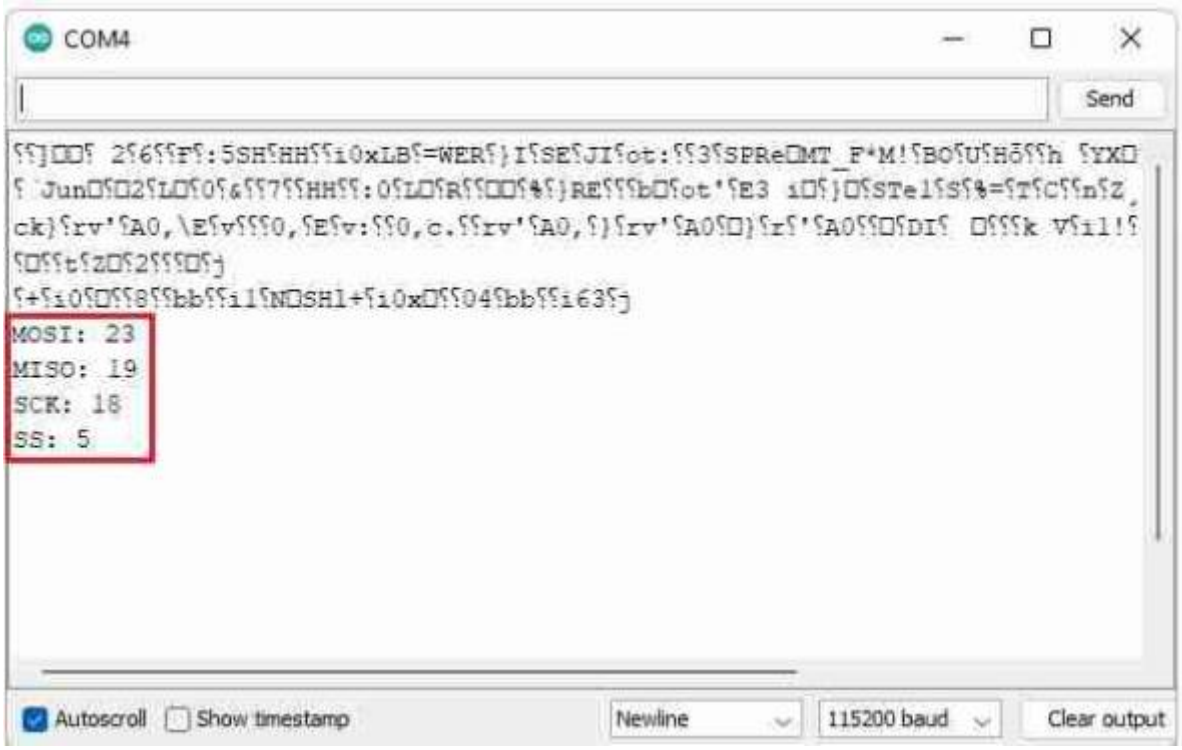
```
  The above copyright notice and this permission notice shall be
included in all
  copies or substantial portions of the Software.
*/

//Find the default SPI pins for your board
//Make sure you have the right board selected in Tools > Boards
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.print("MOSI: ");
  Serial.println(MOSI);
  Serial.print("MISO: ");
  Serial.println(MISO);
  Serial.print("SCK: ");
  Serial.println(SCK);
  Serial.print("SS: ");
  Serial.println(SS);
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Important : assurez-vous de sélectionner la carte que vous utilisez dans Outils > carte , sinon vous risquez de ne pas obtenir les bonnes broches.

Après avoir téléchargé le code, ouvrez le moniteur série, RST votre carte et vous verrez les broches SPI.



Utilisation de broches ESP32 SPI personnalisées

Lorsque vous utilisez des bibliothèques pour vous interfacer avec vos périphériques SPI, il est généralement simple d'utiliser des broches SPI personnalisées car vous pouvez les transmettre en tant qu'arguments au constructeur de la bibliothèque.

Par exemple, jetez un coup d'œil à l'exemple suivant qui s'interface avec un capteur BME280 en utilisant la bibliothèque. **Adafruit_BME280**

bibliothequeadafruitexempleOO1.ino

```
/*
  Rui Santos
  Détails complets du projet sur
https://RandomNerdTutorials.com/esp32-spi-communication-arduino/
  Basé sur l'exemple Adafruit_BME280_Library :
https://github.com/adafruit/Adafruit_BME280_Library/blob/master/example
s/bme280test/bme280test.ino

  Permission est accordée, sans frais, à toute personne obtenant une
copie
  de ce logiciel et des fichiers de documentation associés.

  L'avis de droit d'auteur ci-dessus et cet avis d'autorisation doivent
être inclus dans tous
  des copies ou des parties substantielles du Logiciel.
*/

#comprendre

#comprendre


#comprendre



#comprendre


#define BME_SCK 25
#define BME_MISO 32
#define BME_MOSI 26
#define BME_CS 33
#define SEALEVELPRESSURE_HPA (1013.25)

//Adafruit_BME280bme ; // I2C
//Adafruit_BME280 bme(BME_CS); // SPI matériel
```

```
Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // logiciel
SPI

délai long non signé ;

void setup() {
  Série.begin(9600);
  Serial.println(F("BME280 test"));

  état booléen ;

  // paramètres par défaut
  // (vous pouvez également passer un objet de bibliothèque Wire comme
&Wire2)
  status = bme.begin();
  si (!statut) {
    Serial.println("Impossible de trouver un capteur BME280 valide,
vérifiez le câblage !");
    tandis que (1);
  }

  Serial.println("-- Test par défaut --");
  delayTime = 1000 ;

  Serial.println();
}


boucle vide() {
  printValeurs();
  retard(delayTime);
}

void printValues() {
  Serial.print("Température = ");
  Serial.print(bme.readTemperature());
  Serial.println(" *C");

  // Convertir la température en Fahrenheit
  /*Serial.print("Temperature = ");
  Serial.print(1.8 * bme.readTemperature() + 32);
  Serial.println(" *F");*/

  Serial.print("Pression = ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");

  Serial.print("Altitude approximative = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");
```

```
  Serial.print("Humidité = ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");

  Serial.println();
}
```

Vous pouvez facilement transmettre vos broches SPI personnalisées au constructeur de la bibliothèque.

biblio002.ino

```
Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);
```

Dans ce cas, j'utilisais les broches SPI suivantes (pas par défaut) et tout fonctionnait comme prévu :

biblio003.ino

```
#define BME_SCK 25
#define BME_MISO 32
#define BME_MOSI 26
#define BME_CS 33
```

Si vous n'utilisez pas de bibliothèque, ou si la bibliothèque que vous utilisez n'accepte pas les broches du constructeur de bibliothèque, vous devrez peut-être initialiser le bus SPI vous-même. Dans ce cas, vous devrez appeler leSPI.begin()méthode sur lamettre en place()et passez les broches SPI en arguments :

bliblio004.ino

```
SPI.begin(SCK, MISO, MOSI, SS);
```

Vous pouvez voir un exemple de ce scénario dans ce tutoriel , dans lequel nous initialisons un émetteur-récepteur SPI LoRa qui est connecté à des broches SPI personnalisées. Ou cet exemple montrant comment utiliser des broches SPI personnalisées avec un module de carte microSD .

Liens web

ESP32 SPI autres exemples

broches GPIO ESP32 references