Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Systèmes Embarqués

TP n°1

Programmation avec Arduino

Utilisation de l'ESP32 c3

Vous rajouterez à la fin de votre fichier `~/.bashrc` :

```
export PATH=$HOME/.local/bin:$HOME/bin:$PATH
```

Pour activer la recherche des commandes avec les nouveaux chemins, vous devrez relancer votre terminal.

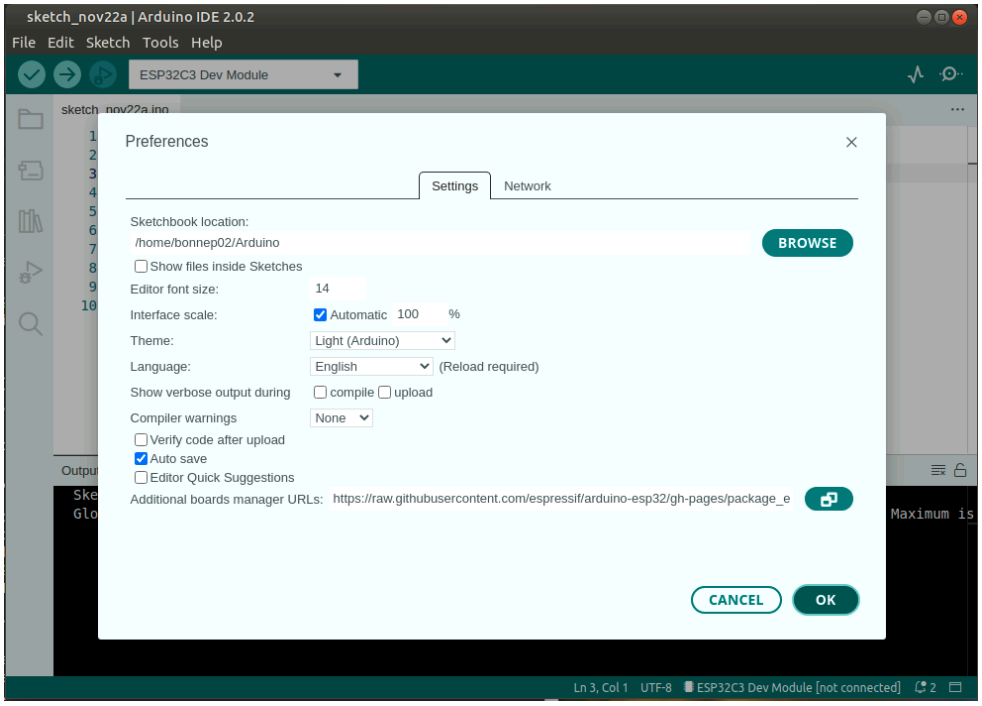
Vous pourrez faire un lien symbolique vers l'exécutable de l'Arduino IDE :

xterm
```
$ ln -s /home/bonnep02/Public/arduino-ide_2.3.2_Linux_64bit.AppImage ~/bin/arduino
```

Ensuite vous lancerez l'IDE Arduino par :

xterm
```
$ arduino
```

Et vous configurerez l'utilisation de l'ESP32 c3 dans l'IDE :



Dans le champs `Additional boards manager URLs`, vous ajouterez la ligne :

```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json
```

L'IDE téléchargera automatiquement la description de différentes « *boards* » et le compilateur pour RiscV.

Ensuite, dans le menu « *Tools* », vous sélectionnerez le sous menu « *Boards Manager* » accessible par la sélection d'une « *board* » différente.
Vous chercherez ESP32 et vous choisirez celui offert par la société Espressif.

Il est possible d'accéder directement à ce panneau en utilisant les icones réparties verticalement à gauche de la fenêtre de l'IDE.

Attention

La carte, ou « *devboard* », à sélectionner est la « *ESP32C3 Dev module* ».

Broches GPIO

1 – Récupérez le « *schematic* » de la carte de développement, « *devboard* », de Muse Lab sur
`https://github.com/wuxx/nanoESP32-C3/blob/master/schematic/nanoESP32C3-v1.0.pdf`

a. Trouvez la broche correspondant à :
- la LED RGB, de type WS2812 ;
- le bouton "BOOT" ;

b. D'après le « *schematic* », le bouton est-il :
- « *active low* », c-à-d quand on presse le bouton l'état logique est 0 ?
- « *active high* », c-à-d quand on presse le bouton l'état logique est 1 ?

Expliquez ce que cela veut dire par rapport au états électriques.

2 – a. Vous testerez le programme suivant :

```
const int buttonPin = ;  // the number of the pushbutton pin
int buttonState = 0;  // variable for reading the pushbutton status

void setup() {
  Serial.begin(115200);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState == LOW) {
    Serial.println("L");
  }
}
```

Vous mettrez le numéro de la broche que vous avez trouvée

Que fait-il ?
Quel rapport avec la petite LED rouge présente sur la carte ?
Si vous changez l'initialisation de la broche de `INPUT` à INPUT_PULLUP, que se passe-t-il ?
Expliquez le comportement.

b. Modifiez le programme précédent pour qu'il n'affiche qu'**une seule fois** le message lorsque l'on appuie sur le bouton, c-à-d que si on maintient appuyé le bouton, un seul affichage se fait.
Est-ce que cela marche comme vous le voulez ?
Pourquoi ?
Est-ce que votre programme détecte un changement d'état sur la broche associée au bouton ?

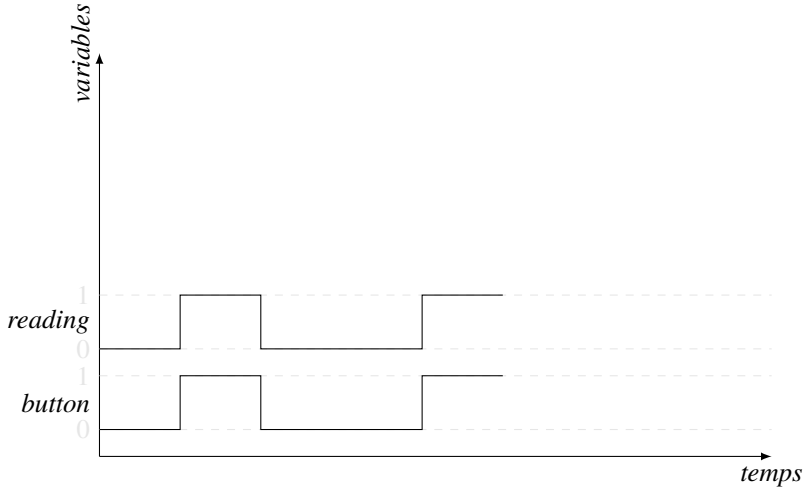
c. Dans les exemples fournis par l'IDE, chargez « *File>Examples>02.Digital>Debounce* ».

Expliquez ce qu'il fait ?

Qu'est-ce que le « *debounce* » d'un bouton ?

Modifiez le **programme précédent** pour intégrer le « *debounce* ».

Modélisez dans un chronogramme qu'est-ce que deviennent les variables du programme lorsque l'on appuie sur le bouton.



IRQ

3 – Vous essaierez le programme suivant :

```
#include <Arduino.h>

struct Button {
    const uint8_t PIN;
    uint32_t numberKeyPresses;
    bool pressed;
};

Button button1 = {9, 0, false};

void ARDUINO_ISR_ATTR isr(void* arg) {
    Button* s = static_cast<Button*>(arg);
    s->numberKeyPresses += 1;
    s->pressed = true;
}

void setup() {
    Serial.begin(115200);
    pinMode(button1.PIN, INPUT_PULLUP);
    attachInterruptArg(digitalPinToInterrupt(button1.PIN), isr, &button1, RISING);
}

void loop() {
    if (button1.pressed) {
        Serial.printf("Button 1 has been pressed %u times\n", button1.numberKey
Presses);
        button1.pressed = false;
    }
}
```

a. Pourquoi met-on `RISING` et non pas `FALLING` ?

b. Que se passe-t-il si vous déplacez l'affichage du compteur d'appui dans la fonction `isr` ? Pourquoi ?

4 – a. Comment fonctionne une LED RGB WS2812 ?

`https://www.sdiplight.com/what-is-ws2812b-led-and-how-to-use-ws2812b-led/`

Vous installerez la bibliothèque permettant de piloter la LED WS2812 de Freenove.

```
#include "Freenove_WS2812_Lib_for_ESP32.h"

#define LEDS_COUNT  1
#define LEDS_PIN    .------ Vous mettrez la broche que vous avez trouvée
#define CHANNEL      0

Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL,
TYPE_GRB);

void setup() {
  strip.begin();
  strip.setBrightness(20);
}

void loop() {
  for (int j = 0; j < 255; j += 2) {
    for (int i = 0; i < LEDS_COUNT; i++) {
      strip.setLedColorData(i, strip.Wheel((i * 256 / LEDS_COUNT + j) & 255));
    }
    strip.show();
    delay(10);
  }
}
```

b. Vous combinerez ce programme avec le précédent pour bloquer l'arc-en-ciel lorsque l'on appui sur un bouton avec un traitement par IRQ.

Timers

5 – Vous essaierez le code suivant :

```
#include "esp_system.h"
hw_timer_t *timer = NULL;
bool tic = false;
int caracteres = 0;

void ARDUINO_ISR_ATTR toc() {
    tic = true;
}

void setup() {
    Serial.begin(115200);
  timer = timerBegin(1000000);
  timerAttachInterrupt(timer, &toc);
  timerAlarm(timer, 1000000, true, 0);
}

void loop() {
    if (tic == true)
  {
        Serial.printf("*");
    tic = false;
    if (++caracteres == 40)
    {
            caracteres = 0;
      Serial.println();
    }
  }
}
```

a. À quelle vitesse s'effectue l'affichage ?
En quelle unité est défini le « *timer* » ?

b. À l'aide de la fonction `millis()` affichez le temps mesurez entre chaque « *tic* » du timer.

c. Que se passe-t-il :
   - ◇ si vous essayez de diminuer le temps de déclenchement du timer ?
   - ◇ si vous mettez le calcul et l'affichage de la durée dans la fonction `tick` ?

d. Soit le code suivant :

```
portMUX_TYPE m = portMUX_INITIALIZER_UNLOCKED;

void IRAM_ATTR toc()
{
portENTER_CRITICAL_ISR(&m);
...
portEXIT_CRITICAL_ISR(&m);
}
```

Que fait le code ?
À quoi sert ❶ ?
Qu'est-ce que veut dire ❷ ?
Reprenez votre code pour intégrer cette opération.

Si vous n'arrivez pas à programmer votre ESP32

1. Maintenir appuyé le bouton "BOOT" ;
2. Appuyer le bouton "RST" ;
3. Relâcher le bouton "BOOT" ;

`https://docs.espressif.com/projects/esptool/en/latest/esp32c3/advanced-topics/boot-mode-selection.html`