Programmation FreeRTOS

Installation de l'environnement de développement Arduino

Vous installerez l'IDE Arduino en version 1.8.19 disponible à `https://www.arduino.cc/`.

xterm

```
$ wget https://downloads.arduino.cc/arduino-1.8.19-linux64.tar.xz
```

Vous installerez ensuite l'environnement de développement pour l'ESP32 ou l'ESP8266 dans l'IDE Arduino :

- dans « *Préférences* », vous sélectionnerez « *Additional Boards Manager URLs* » ;
- vous ajouterez :

  ESP8266 `https://arduino.esp8266.com/stable/package_esp8266com_index.json`

  ESP32 `https://dl.espressif.com/dl/package_esp32_index.json`

Programmation freeRTOS : lancement de tâches et priorité

```
#include <stdlib.h>

// Use only core 1 for demo purposes
#if CONFIG_FREERTOS_UNICORE
  static const BaseType_t app_cpu = 0;
#else
  static const BaseType_t app_cpu = 1;
#endif
// Pins
static const int led_pin = LED_BUILTIN;
// Globals
static int led_delay = 500;   // ms

// Task: Blink LED at rate set by global variable
void toggleLED(void *parameter) {
  while (1) {
    digitalWrite(led_pin, HIGH);
    vTaskDelay(led_delay / portTICK_PERIOD_MS);
    digitalWrite(led_pin, LOW);
    vTaskDelay(led_delay / portTICK_PERIOD_MS);
  }
}

void setup() {
  // Configure pin
  pinMode(led_pin, OUTPUT);

  // Configure serial and wait a second
  Serial.begin(115200);
  vTaskDelay(1000 / portTICK_PERIOD_MS);
  Serial.println("freeRTOS LED Demo");

  // Start blink task
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
            toggleLED,      // Function to be called
            "Toggle LED",   // Name of task
            1024,           // Stack size (bytes in ESP32, words in FreeRTOS)
            NULL,           // Parameter to pass
            1,              // Task priority
            NULL,           // Task handle
            app_cpu);       // Run on one core for demo purposes (ESP32 only)
  // En freeRTOS classique il faudrait appeler vTaskStartScheduler()
  // pour débuter l'ordonnanceur, ici avec Arduino c'est fait automatiquement
}

void loop() {
  // Execution should never get here
}
```

Travail

Écrire un programme qui ajoute une nouvelle tâche qui :
- ▷ lit le port série dans cette nouvelle tâche pour trouver une valeur de délai ;
- ▷ fait clignoter la LED avec la valeur lue.

Pour lire le port série :

```
void readSerial(void *parameters) {

  char c;
  char buf[buf_len];
  uint8_t idx = 0;

  // Clear whole buffer
  memset(buf, 0, buf_len);

  // Loop forever
  while (1) {

    // Read characters from serial
    if (Serial.available() > 0) {
      c = Serial.read();

      // Update delay variable and reset buffer if we get a newline character
      if (c == '\n') {
        led_delay = atoi(buf);
        Serial.print("Updated LED delay to: ");
        Serial.println(led_delay);
        memset(buf, 0, buf_len);
        idx = 0;
      } else {

        // Only append if index is not over message limit
        if (idx < buf_len - 1) {
          buf[idx] = c;
          idx++;
        }
      }
    }
  }
}
```