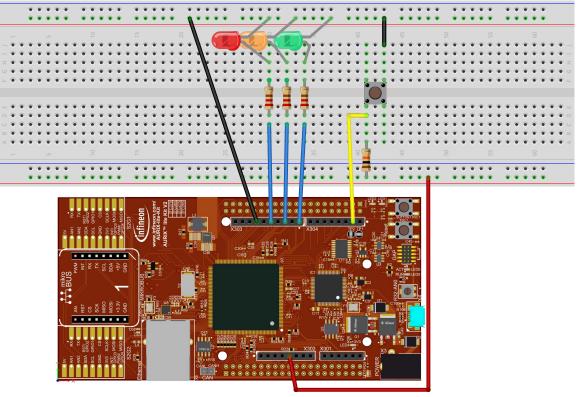
Übung 2-3 – Echtzeitbetriebssysteme

FreeRTOS

In dieser Aufgabe soll das Task-Verhalten aus Übung 2-2 mithilfe des Preemptiven Echtzeitbetriebssystems **FreeRTOS** umgesetzt werden. Verwenden Sie dafür den Hardwareaufbau aus dem vorherigen Seminar und ergänzen diesen durch einen zusätzlichen Taster am Eingang 2 (siehe Bild).



fritzing

In dieser Übung sollen die drei LED – die jeweils **durch eine eigene Task** angesteuert werden – ebenfalls mit unterschiedlicher Frequenz blinken. Zusätzlich soll die Frequenz der LED 1 durch einen Tastendruck verändert werden können. Das **Überwachen des Tasters** erfolgt über einen Interrupt und **einer vierten Task**.

Vorbereitungen:

- Öffnen Sie den Template (siehe OPAL).
- Die Bibliothek *FreeRTOS* ist bereits im Template eingebunden. Die Quellcode befindet sich im **OS** Folder.

Implementierung:

- Erzeugen zu Beginn der main-Funktion eine Queue mit 5 Elementen der Größe 2 Byte (uint16_t) mittels der Funktion xQueueCreate(...) (siehe Seite 3)
- Erzeugen im Weiteren vier Tasks:

```
Task_L1 (Prio 2)
Task_L2 (Prio 2)
Task_L3 (Prio 2)
Task_Bt (Prio 1)
mit der Funktion xTaskCreate(...) (siehe Seite 3)
```

- Starten Sie gleich vor dem while Schleife den Scheduler mittels vTaskStartScheduler();
- Definieren Sie die entsprechenden drei Task-Funktionen für die LEDs, welche in einer Endlosschleife die LED blinken lassen
- Definieren Sie eine Task-Funktion zum überwachen des Buttons, wobei bei einem erfolgten Tastendruck eine Nachricht (ohne Blockierzeit) in die Queue gelegt werden (Funktion: xQueueSend(...)) (siehe Seite 3) Dabei soll als Nachricht die Intervallzeit wechselseitig 100 (5 Hz) und 500 (1 Hz) übertragen werden
- Erweitern Sie die Task der LED 1 um das Lesen aus der Queue (ohne Blockierzeit) mittels der Funktion **xQueueReceive(...)** (siehe Seite 3) Übernehmen Sie den Wert der Nachricht als neue Intervallzeit für das Blinken der LED

Erzeugen einer Task:

```
BaseType_t xTaskCreate(
    TaskFunction_t pvTaskCode, // z.B. Task_L1
    const char * const pcName, // z.B. "Led 1 blinking"
    uint16_t usStackDepth, // z.B. 128 (Stacksize)
    void *pvParameters, // z.B. NULL (no params)
    UBaseType_t uxPriority, // z.B. 2 (Prio)
    TaskHandle_t *pxCreatedTask); // z.B. NULL
    // (Task Handle not used)
```

Erzeugen einer Queue:

```
QueueHandle_t xQueueCreate(

UBaseType_t uxQueueLength, // z.B. 5

UBaseType_t uxItemSize ); // z.B. 2
```

Senden einer Nachricht:

```
BaseType_t xQueueSend(
QueueHandle_t xQueue, // myQueue

const void * pvItemToQueue, // z.B. &data
TickType_t xTicksToWait ); // 0
```

Empfangen einer Nachricht: