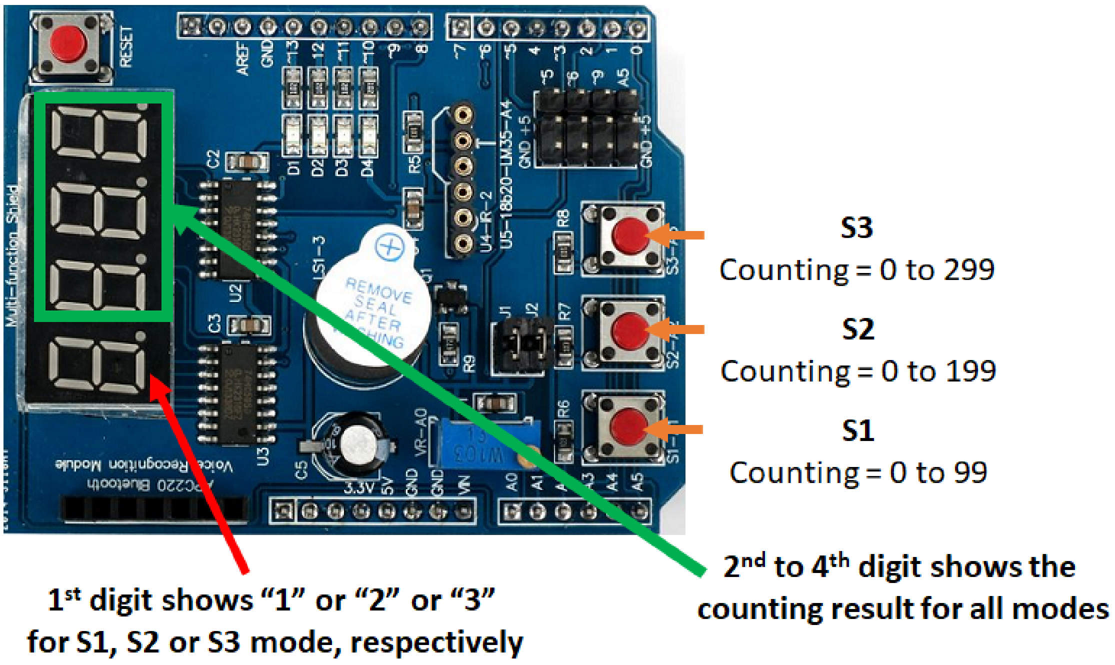
Pergunta 1

Por responder Pontuação 26,000

Consider a *microcontroller unit (MCU)* to develop a program for counting seconds. This MCU needs to display the accumulated time when the button S1 or S2 or S3 is pressed (and released), to a maximum duration of 100, 200 or 300 seconds, respectively. Use the Arduino board and the MultiFunction Shield of the previous lectures along with the FreeRTOS library to complete the following code. See figure below for more details.


S3
Counting = 0 to 299
S2
Counting = 0 to 199
S1
Counting = 0 to 99
2nd to 4th digit shows the counting result for all modes
1st digit shows "1" or "2" or "3" for S1, S2 or S3 mode, respectively

Complete the code which needs to guarantee that the timing of each mode is presented correctly, meaning that:

1. All modes are available at the start of the MCU
2. Only **one mode** must be functioning and displaying the time, even if several buttons are pressed at the same time.
3. A mode will start and finish counting the time **without being interrupted** however, the MCU must be responsive to other events.
4. After concluding a specific mode, the MCU needs to **automatically start any other** mode whose button was pressed before.
5. In case of multiple buttons being pressed altogether: the running mode must terminate and after that, the **mode S1 needs to start always if button S1** was pressed. There is **no particular sequence of operation for other modes** (S2 and S3).

According to the previous requirements, please complete the following program with the correct answers:

```
#include <Arduino_FreeRTOS.h>
#include <MultiFunctionShield.h>
#include <semphr.h>
#define TIME_UPDATE 100
MultiFunctionShield MFS;

TaskHandle_t Handle_S1;
TaskHandle_t Handle_S2;
TaskHandle_t Handle_S3;
SemaphoreHandle_t mutex;


void TaskReadInputs( void *pvParameters );
void TaskCount100( void *pvParameters );
void TaskCount200 ( void *pvParameters );
void TaskCount300( void *pvParameters );


void setup() {
  MFS.begin();
  mutex = xSemaphoreCreateMutex();
  xTaskCreate(TaskReadInputs, "Task_Inputs", 128, NULL,          ,          );

  xTaskCreate(TaskCount100, "Task_Read1_S1", 128, NULL,          ,          );

  xTaskCreate(TaskCount200, "Task_Read_S2", 128, NULL,          ,          );
  xTaskCreate(TaskCount300, "Task_Read1_S3", 128, NULL,            ,                  );
  vTaskStartScheduler();
}
void displayValue(int count, int button) {
  for(int i = 0; i < count; i++){
     delay(TIME_UPDATE);
     MFS.Display(i+button*1000);
  }
  return;
}
void TaskCount100(void *pvParameters) {
  while(1) {
    int count = 100;
    if                                                  {
     displayValue(count, 1);



    }
  }
}
void TaskCount200(void *pvParameters) {
  while(1) {
    int count = 200;
    if                                                  {
     displayValue(count, 2);



    }
  }
}
void TaskCount300(void *pvParameters) {
  while(1) {
    int count = 300;
    if                                                  {
     displayValue(count, 3);



    }
  }
}
void TaskReadInputs(void *pvParameters) {
  pinMode(BUTTON_1_PIN, INPUT);
  pinMode(BUTTON_2_PIN, INPUT);
  pinMode(BUTTON_3_PIN, INPUT);
  delay(1000);

  vTaskSuspend(                  );
  vTaskSuspend(                  );
  vTaskSuspend(                  );
  while(1){
    if(                            < 10)

    if(                          < 10)


    if(                          < 10)


    vTaskDelay (100/ portTICK_PERIOD_MS);
  }
}
void loop() {}
```