FACULDADE: CENTRO UNIVERSITÁRIO DE BRASÍLIA – EDUCAÇÃO SUPERIOR
CURSO: ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA: SISTEMAS EMBARCADOS
CARGA HORÁRIA: 60 H. A. **ANO/SEMESTRE:** 2021/02
PROFESSOR: ADERBAL BOTELHO
ALUNOS: MATHEUS BARCELOS DE CARVALHO (21907159) E EDUARDO AFONSO DA SILVA INÁCIO (21908507)

LABORATÓRIO – SINCRONIZAÇÃO E COMUNICAÇÃO

BRASÍLIA,
17 DE AGOSTO DE 2021.

Contribuição no trabalho

- Na parte prática do trabalho, o Matheus ficou responsável pelos exercícios 1 e 2 e o Eduardo pelo exercício 3.
- Já a parte teórica, foi feita em conjunto como um todo.

Introdução

Sistemas de tempo real são extremamente importantes para a segurança e o funcionamento correto de diversos sistemas embarcados utilizados na atualidade. A grande diferença dos sistemas de tempo real para os sistemas em geral, é que os sistemas em tempo real executam o trabalho no tempo disponível, já os outros executam o trabalho no tempo que precisam. Os sistemas de tempo real podem ser encontrados em diversos ambientes, como por exemplo: na automação de naves, radares ou vídeo games.

Sistemas Embarcados

Sistemas embarcados são sistemas computacionais que utilizam um microcontrolador (podendo ter a utilização ou não de sistemas operacionais), e diferente dos computadores de propósito geral, servem exclusivamente para realizar uma tarefa específica, tendo como exemplos: medir a temperatura de um ambiente, acender leds de acordo com pré especificações e etc. Através da engenharia, é possível otimizar esse sistema através da diminuição do código para execução, da redução de tamanho físico do projeto ou diminuindo o uso de recursos computacionais.

Sistemas Operacionais

O sistema operacional é um software ou um conjunto de softwares que são responsáveis por gerenciar e administrar os recursos do computador, ele é quem carrega informações, faz a conexão entre o hardware e o software e cria a interface para que possamos nos “comunicar” com os mesmos.

O sistema operacional introduz uma “camada de abstração” entre o hardware e o usuário, transformando, por exemplo, comandos do mouse ou teclado e solicitações do sistema, como gerenciamento de recursos (CPU, memória RAM), em linguagem de máquina, assim enviando instruções ao processador. O último traduz estas instruções para código binário, executa os comandos e envia as informações para a interface.

Um sistema operacional é dividido entre bibliotecas, programas, interface e as instruções que compõem o seu núcleo (kernel).

Sistemas Operacionais de tempo real

Já os sistemas operacionais de tempo real, ou RTOS, são sistemas operacionais destinados a tarefas em que é essencial a confiabilidade e a execução de tarefas com prazos compatíveis a acontecimentos externos, tendo como um exemplo o airbag de um carro, que precisa de uma alta prioridade e um tempo de execução limite para que não haja consequências críticas em um acidente de carro. Outro exemplo a ser dado é quando um avião em piloto automático é desviado de sua trajetória, o controle do avião deve corrigir imediatamente a rota.

Nesse sistema, é dado um prazo limite pré-determinado para execução e há uma liberdade maior para o desenvolvedor definir as prioridades para realização de cada tarefa, onde de acordo com a prioridade, ela pode ser interrompida para a execução de outra. As tarefas são realizadas individualmente(quando uma está sendo executada, a outra está em pausa), porém, elas são executadas rapidamente de acordo com a prioridade, de maneira alternada, dando a impressão de uma realização de multitarefas. Esse efeito multitarefas também ocorre nos sistemas operacionais

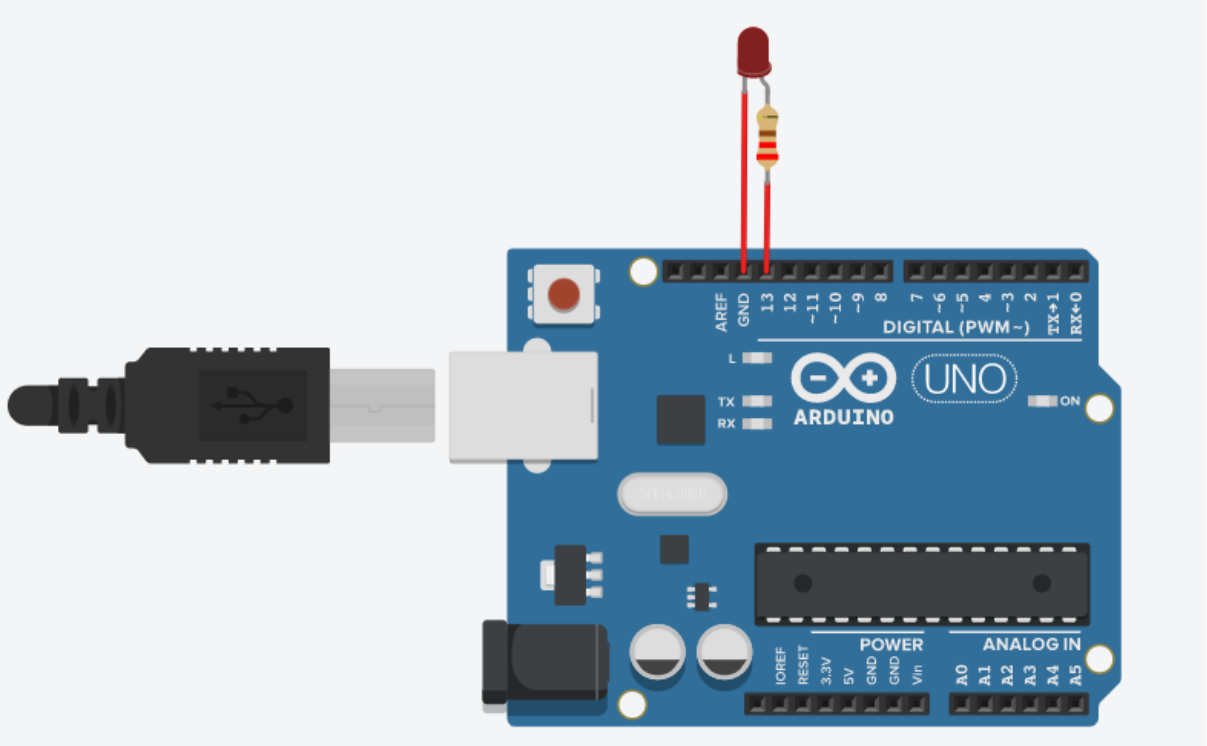
comuns, porém, no caso de um sistema de tempo real, se uma tarefa não for realizada dentro de seu prazo pré-determinado, há uma falha no sistema, o que não ocorre nos sistemas operacionais de propósito geral.

Exercícios

Exercício - 1 e 2

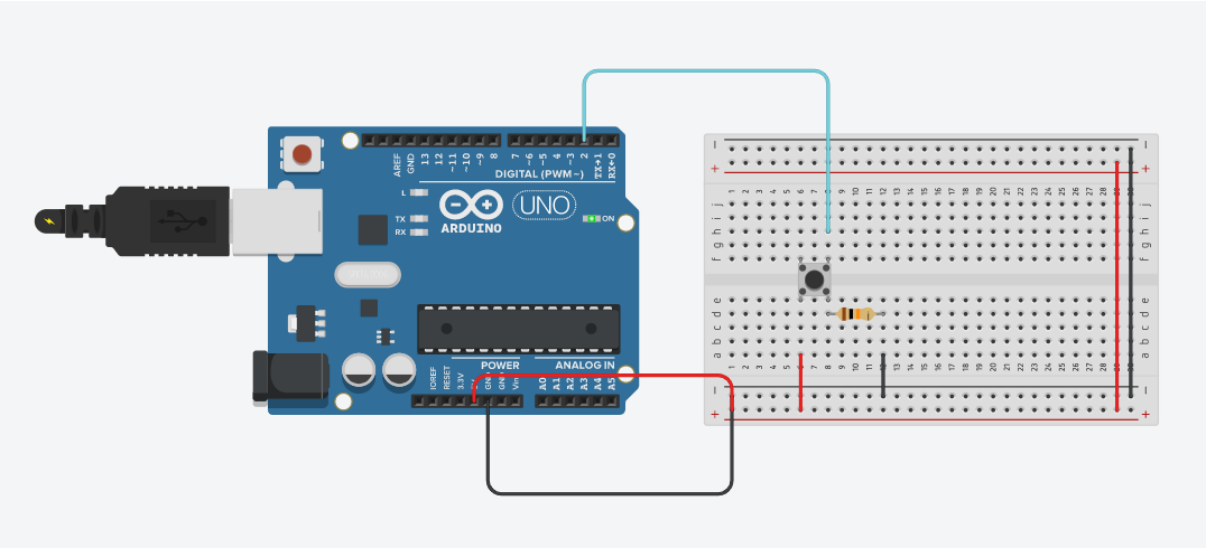
Os dois primeiros exercícios são bem simples. Ambos são programados da forma bare-metal, ou seja, diretamente com o arduíno, sem utilização de sistemas operacionais. Desta forma, é utilizado o super loop, onde todas as informações são colocadas na função “main” e rodadas em loop infinito, sendo assim, as tasks são executadas uma atrás da outra.

A primeira questão consiste em fazer um LED piscar, já a segunda, imprimir no monitor serial o estado do botão.

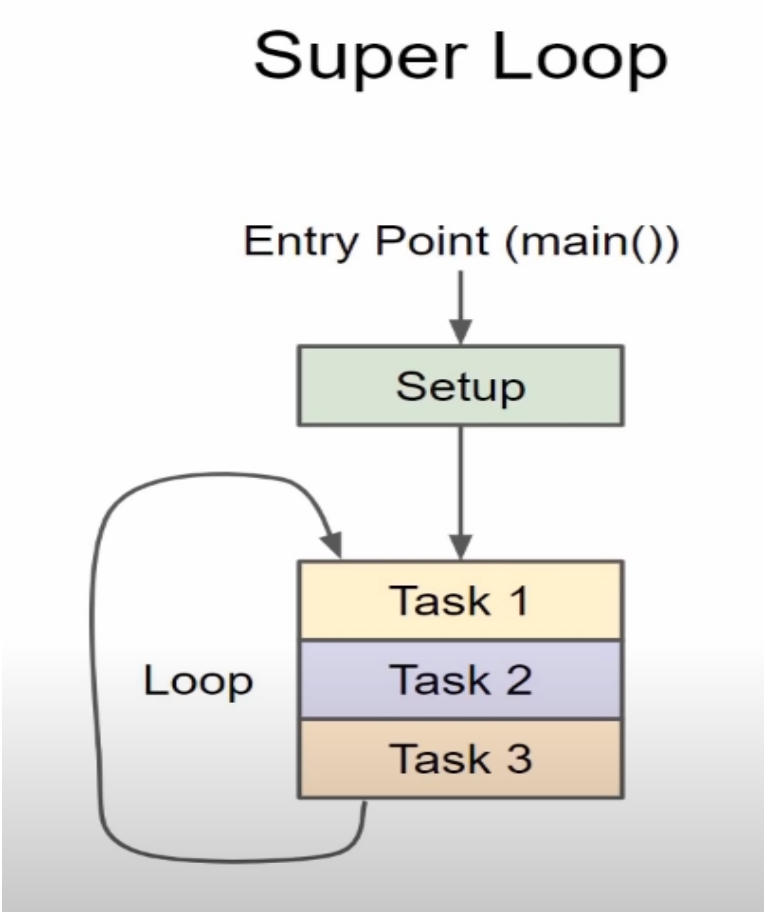


```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(500);
  digitalWrite(13, LOW);
  delay(500);
}
```

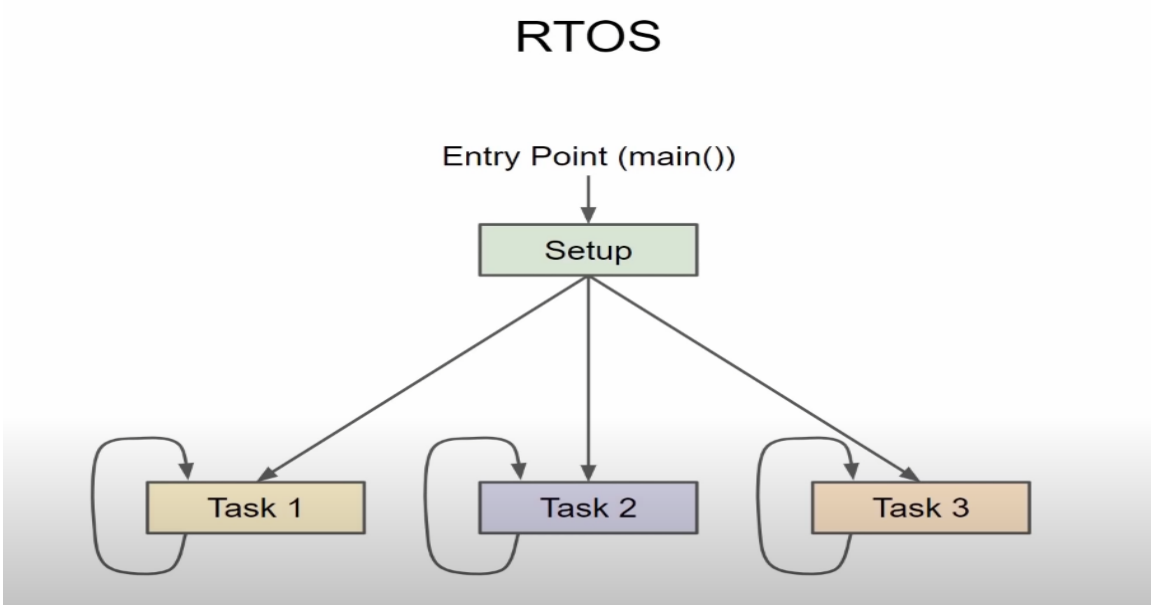


```
/*
DigitalReadSerial
Reads a digital input on pin 2, prints the result to the serial r
This example code is in the public domain.
*/
// digital pin 2 has a pushbutton attached to it. Give it a name
int pushButton = 2;
// the setup routine runs once when you press reset:
void setup() {
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
// make the pushbutton's pin an input:
pinMode(pushButton, INPUT);
}
// the loop routine runs over and over again forever:
void loop() {
// read the input pin:
int buttonState = digitalRead(pushButton);
// print out the state of the button:
Serial.println(buttonState);
delay(1); // delay in between reads for stability
}
```



Exercício - 3

Nesse caso, a programação é feita utilizando um sistema operacional em tempo real, logo, é dividida em tasks, para que todas sejam executadas “ao mesmo tempo”(multitarefas).



O exercício 3 consiste basicamente na utilização de um sistema operacional em tempo real (NilRTOS) para criar um semáforo. Ao haver 2 tasks sendo executadas ‘’ao mesmo tempo’’, uma pode acabar utilizando de um recurso que a outra já está utilizando. Logo, a utilidade do semáforo é indicar se o recurso já está sendo utilizado para evitar problemas maiores. As tarefas podem utilizar o semáforo para fazer essa verificação, assim, continuar sua execução apenas quando for possível.

```
/*
* Example to demonstrate thread definition, semaphores, and thread sleep.
*/
#include <NilRTOS.h>
// The LED is attached to pin 13 on Arduino.
const uint8_t LED_PIN = 13;
// Declare a semaphore with an inital counter value of zero.
SEMAPHORE_DECL(sem, 0);
//------------------------------------------------------------------------------
/*
* Thread 1, turn the LED off when signalled by thread 2.
*/
// Declare a stack with 128 bytes beyond context switch and interrupt needs.
NIL_WORKING_AREA(waThread1, 128);
// Declare the thread function for thread 1.
NIL_THREAD(Thread1, arg) {
while (TRUE) {
// Wait for signal from thread 2.
nilSemWait(&sem);
// Turn LED off.
digitalWrite(LED_PIN, LOW);
}
}
//------------------------------------------------------------------------------
/*
* Thread 2, turn the LED on and signal thread 1 to turn the LED off.
*/
// Declare a stack with 128 bytes beyond context switch and interrupt needs.
NIL_WORKING_AREA(waThread2, 128);
// Declare the thread function for thread 2.
NIL_THREAD(Thread2, arg) {
pinMode(LED_PIN, OUTPUT);
while (TRUE) {
  // Turn LED on.
digitalWrite(LED_PIN, HIGH);
// Sleep for 200 milliseconds.
nilThdSleepMilliseconds(200);
// Signal thread 1 to turn LED off.
nilSemSignal(&sem);
// Sleep for 200 milliseconds.
nilThdSleepMilliseconds(200);
}
}

/*
* Threads static table, one entry per thread. A thread's priority is
* determined by its position in the table with highest priority first.
*
* These threads start with a null argument. A thread's name may also
* be null to save RAM since the name is currently not used.
*/
NIL_THREADS_TABLE_BEGIN()
NIL_THREADS_TABLE_ENTRY("thread1", Thread1, NULL, waThread1, sizeof(waThread1))
NIL_THREADS_TABLE_ENTRY("thread2", Thread2, NULL, waThread2, sizeof(waThread2))
NIL_THREADS_TABLE_END()
//------------------------------------------------------------------------------
void setup() {
// Start Nil RTOS.
nilSysBegin();
}
//------------------------------------------------------------------------------
// Loop is the idle thread. The idle thread must not invoke any
// kernel primitive able to change its state to not runnable.
void loop() {
// Not used.
}
```

Bibliografia

O QUE É UM SISTEMA operacional?: Aprenda o que é um sistema operacional, o que ele faz em seu computador ou celular e entenda a diferença entre kernel e firmware. 2021. Disponível em: https://tecnoblog.net/303055/o-que-e-um-sistema-operacional/. Acesso em: 17 ago. 2021.

SISTEMA embarcado. 2021. Disponível em: https://pt.wikipedia.org/wiki/Sistema_embarcado. Acesso em: 17 ago. 2021.

SISTEMA em Tempo Real - SO. [*S. l.*: *s. n.*], 2015. Disponível em: 07/10/2015. Acesso em: 17 ago. 2021.

PALESTRA: MELHORE SEUS PROJETOS ARDUINO USANDO FREERTOS (ARDUINO DAY VIRTUAL 2020 - PEDRO BERTOLETI). [*S. l.*: *s. n.*], 2020. Disponível em: 07/10/2020. Acesso em: 17 ago. 2021.

INTRODUCTION to RTOS Part 1 - What is a Real-Time Operating System (RTOS)? | Digi-Key Electronics. [*S. l.*: *s. n.*], 2021. Disponível em: 04/01/2021. Acesso em: 17 ago. 2021.