Ministry of Education, Culture and Research of the Republic of Moldova

Technical University of Moldova

Department of Software and Automatic Engineering

REPORT

Laboratory Project nr.5
at Embeded Systems

Done by:
Gr.191-FAF Nicolae Basso

Checked by: Andrei Bragarenco

Chișinău 2022

Laboratory Project Nr.5

Topic: Sequential operating systems

Objective: The objective of this laboratory work is to understand how sequential operating systems work and how to implement a simple one on and MCU. To understand how to schedule tasks and manage them.

Domain: An **operating system** (**OS**) is system software that manages computer hardware, software resources, and provides common services for computer programs. Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources. For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware,[1][2] although the application code is usually executed directly by the hardware and frequently makes system calls to an OS function or is interrupted by it. Operating systems are found on many devices that contain a computer – from cellular phones and video game consoles to web servers and supercomputers.

Component description:

74HC595 - The 74HC595 consists of an 8−bit shift register and an 8−bit D−type latch with three−state parallel outputs. The shift register accepts serial data and provides a serial output. The shift register also provides parallel data to the 8−bit latch. The shift register and latch have independent clock inputs. This device also has an asynchronous reset for the shift register.

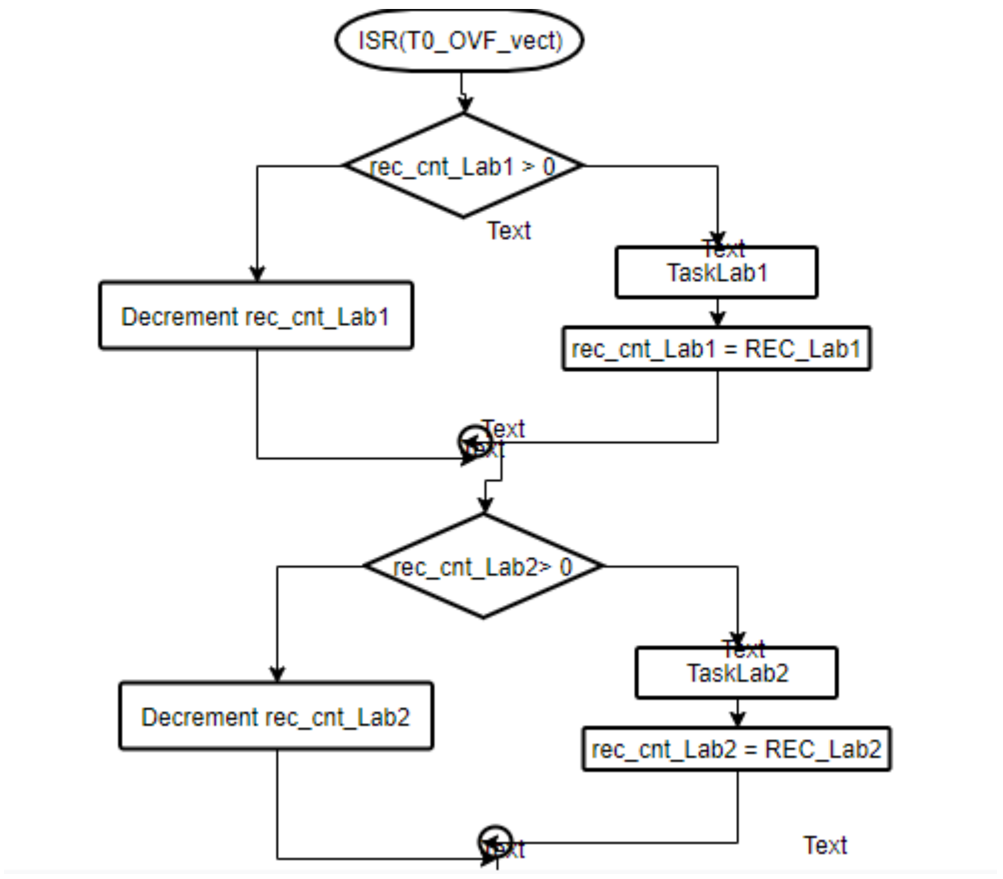
L298 MOTOR DRIVER - The L298 is an integrated monolithic circuit in a 15- lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

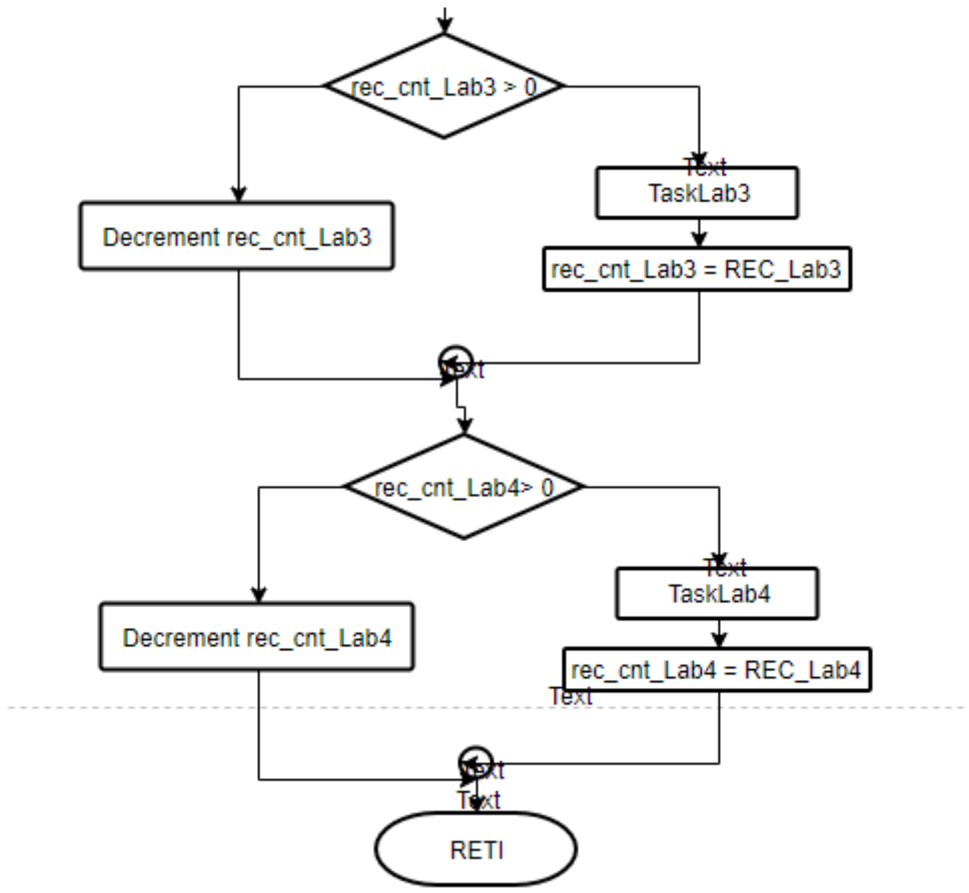
Motor - A **DC motor** is any of a class of rotary electrical **motors** that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields.

Electric lamp - An **electric lamp** is a conventional light emitting component used in different circuits, mainly for lighting and indicating purposes. The construction of lamp is quite simple, it has one filament surrounding which, a transparent glass made spherical cover is provided. The filament of the lamp is mainly made of tungsten as it has high melting point temperature. A lamp emits light energy as the thin small tungsten filament of lamp glows without being melted, while current flows through it.

Arduino Uno - The **Arduino Uno** is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc.[2][3] The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits.[1] The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable.[4] It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo.[5][6] The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

Implementation:





Conclusions: Working on this laboratory work I have understood what operating systems are, what kind on operating systems there are for MCUs: sequential OS and preemptive OS. These two kinds of Oss allow the processor to take a breath while working. By scheduling the tasks we can specify when a specific task should run and so the processor will not be always fully loaded. Therefor the system becomes more responsive by avoiding spin locks. In order to create a simple sequential operating system for Arduino Uno I have user the "timer-api" library which allows to use interrupts for running tasks at a specific time. The library is a Simple cross-platform API for multitasking on Arduino based on timer interrupt handlers

Annex:

```
#include <Arduino.h>
#include "Lab1.h"
#include "Lab2.h"
#include "Lab3.h"
#include "Lab4.h"
#include "timer-api.h"

#define OFS_Lab1 1000
#define REC_Lab1 1000
int rec_cnt_Lab1 = OFS_Lab1;
void TaskLab1() {
    Lab1 lab1;
    lab1.setup();
    lab1.loop();
```

```
}

#define OFS_Lab2 10000
#define REC_Lab2 1000
int rec_cnt_Lab2 = OFS_Lab2;
void TaskLab2() {
    Lab1 lab2;
    lab2.setup();
    lab2.loop();
}

#define OFS_Lab3 100000
#define REC_Lab3 1000
int rec_cnt_Lab3 = OFS_Lab3;
void TaskLab3() {
    Lab1 lab3;
    lab3.setup();
    lab3.loop();
}

#define OFS_Lab4 1000000
#define REC_Lab4 1000
int rec_cnt_Lab4 = OFS_Lab4;
void TaskLab4() {
    Lab1 lab4;
    lab4.setup();
    lab4.loop();
}

void timer_handle_interrupts(int timer) {
    if(--rec_cnt_Lab1 <= 0) {
        TaskLab1();
        rec_cnt_Lab1 = REC_Lab1;
    }
    if(--rec_cnt_Lab2 <= 0) {
        TaskLab2();
        rec_cnt_Lab2 = REC_Lab2;
    }
    if(--rec_cnt_Lab3 <= 0) {
        TaskLab3();
        rec_cnt_Lab3 = REC_Lab3;
    }
    if(--rec_cnt_Lab4 <= 0) {
        TaskLab4();
        rec_cnt_Lab4 = REC_Lab4;
    }
}

void setup() {
    timer_init_ISR_1KHz(TIMER_DEFAULT);
```

```
}

void loop() { }
```