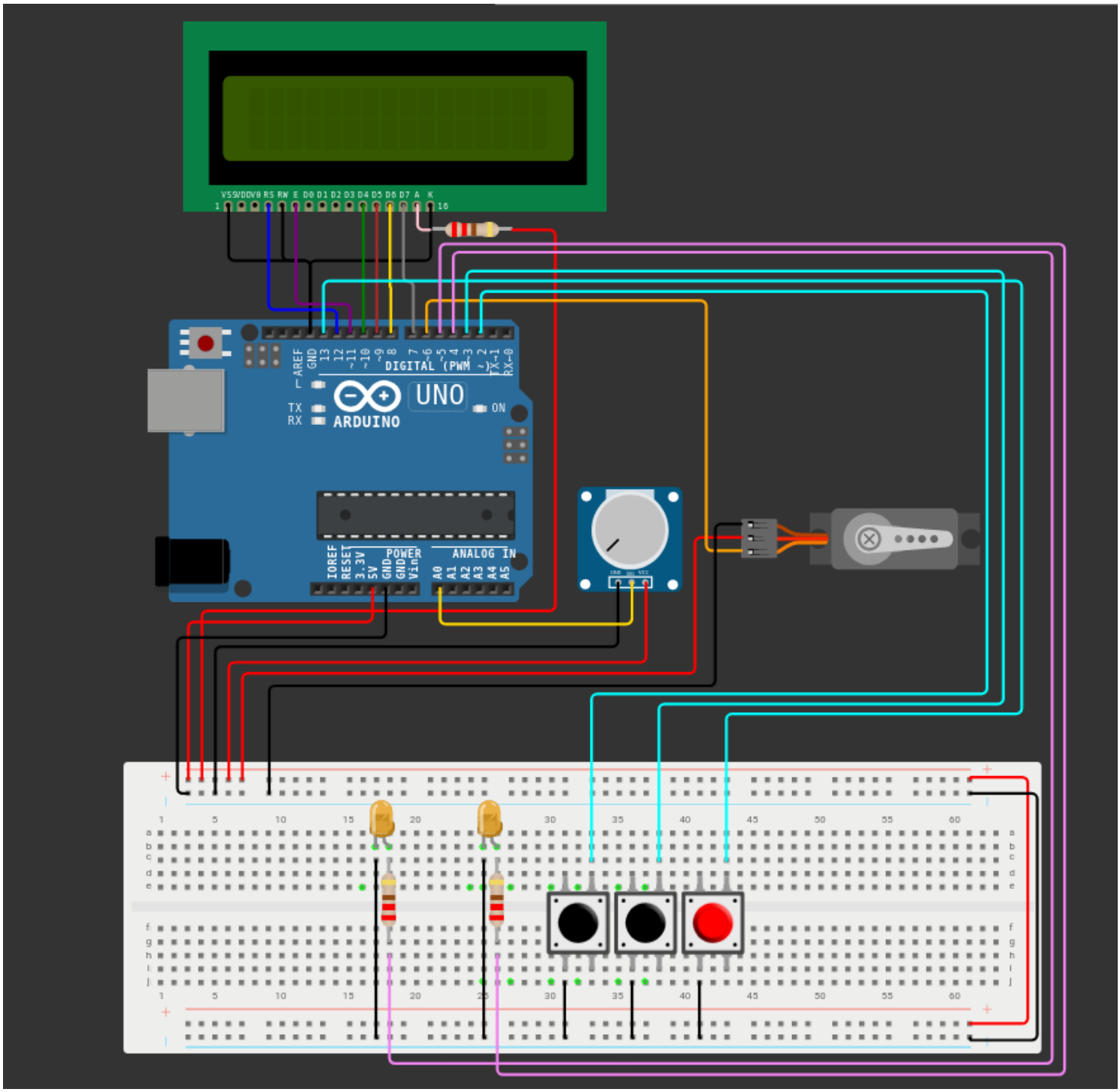
Μικροϋπολογιστές: Εργαστηριακή άσκηση 4

Χρήστος Μαργιώλης – 19390133

Δεκέμβριος 2022

1 Κύκλωμα



2 Κώδικας

```
#include <Arduino_FreeRTOS.h>
#include <LiquidCrystal.h>
#include <Servo.h>
#include <queue.h>

#define PIN_BTN_LEFT 2
#define PIN_BTN_RIGHT 3
#define PIN_BTN_ALARM 13
#define PIN_TURN_LEFT 4
#define PIN_TURN_RIGHT 5
#define PIN_SERVO 6
#define PIN_POTENTIOMETER A0

#define TURN_SIGNAL_DELAY 500

QueueHandle_t queue;
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);
Servo servo;

void
setup()
{
        Serial.begin(9600);
        pinMode(PIN_BTN_LEFT, INPUT_PULLUP);
        pinMode(PIN_BTN_RIGHT, INPUT_PULLUP);
        pinMode(PIN_BTN_ALARM, INPUT_PULLUP);
        lcd.begin(16, 2);
        servo.attach(PIN_SERVO);
        queue = xQueueCreate(5, sizeof(int));
        if (queue == NULL) {
                Serial.println("queue cannot be created");
                /* hang */
                for (;;);
        }
        xTaskCreate(turn_signal_left, "Left turn singal", 128, NULL, 1, NULL);
        xTaskCreate(turn_signal_right, "Right turn singal", 128, NULL, 1, NULL);
        xTaskCreate(alarm, "Alarm", 128, NULL, 1, NULL);
        xTaskCreate(rpm_count, "RPM count", 128, NULL, 1, NULL);
        xTaskCreate(tachometer, "Tachometer", 128, NULL, 1, NULL);
        vTaskStartScheduler();
}

void
loop()
```

```
{
}

int
btn_pressed(int btn_pin)
{
        return (digitalRead(btn_pin) == LOW);
}

void
turn_signal_left(void *pv_params)
{
        if (!btn_pressed(PIN_BTN_LEFT))
                return;
        pinMode(PIN_TURN_LEFT, OUTPUT);
        for (;;) {
                Serial.println("Left turn");
                digitalWrite(PIN_TURN_LEFT, HIGH);
                vTaskDelay(TURN_SIGNAL_DELAY / portTICK_PERIOD_MS);
                digitalWrite(PIN_TURN_LEFT, LOW);
                vTaskDelay(TURN_SIGNAL_DELAY / portTICK_PERIOD_MS);
        }
}

void
turn_signal_right(void *pv_params)
{
        if (!btn_pressed(PIN_BTN_RIGHT))
                return;
        pinMode(PIN_TURN_RIGHT, OUTPUT);
        for (;;) {
                Serial.println("Right turn");
                digitalWrite(PIN_TURN_RIGHT, HIGH);
                vTaskDelay(TURN_SIGNAL_DELAY / portTICK_PERIOD_MS);
                digitalWrite(PIN_TURN_RIGHT, LOW);
                vTaskDelay(TURN_SIGNAL_DELAY / portTICK_PERIOD_MS);
        }
}

void
alarm(void *pv_params)
{
        if (!btn_pressed(PIN_BTN_ALARM))
                return;
        pinMode(PIN_TURN_LEFT, OUTPUT);
        pinMode(PIN_TURN_RIGHT, OUTPUT);
```

```
        for (;;) {
                Serial.println("Alarm");
                digitalWrite(PIN_TURN_LEFT, HIGH);
                digitalWrite(PIN_TURN_RIGHT, HIGH);
                vTaskDelay(TURN_SIGNAL_DELAY / portTICK_PERIOD_MS);
                digitalWrite(PIN_TURN_LEFT, LOW);
                digitalWrite(PIN_TURN_RIGHT, LOW);
                vTaskDelay(TURN_SIGNAL_DELAY / portTICK_PERIOD_MS);
        }
}

void
rpm_count(void *pv_params)
{
        int rpm, servo_angle;

        for (;;) {
                servo_angle = map(analogRead(PIN_POTENTIOMETER), 0, 1023, 0, 180);
                servo.write(servo_angle);
                rpm = map(servo_angle, 0, 180, 0, 8000);
                Serial.print("RPM: ");
                Serial.println(rpm);
                xQueueSend(queue, &rpm, portMAX_DELAY);
                vTaskDelay(1000 / portTICK_PERIOD_MS);
        }
}

void
tachometer(void *pv_params)
{
        int rpm;

        for (;;) {
                if (xQueueReceive(queue, &rpm, portMAX_DELAY) != pdPASS)
                        return;
                Serial.print("Tachometer: ");
                Serial.println(rpm);
                lcd.clear();
                lcd.setCursor(0, 0);
                lcd.print("RPM: ");
                lcd.setCursor(6, 0);
                lcd.print(rpm);
        }
}
```