НП „ИТ КАРИЕРА“
МОДУЛ VIII
РАЗРАБОТКА НА СОФТУЕР

“Таймер Аларма”

Изготвила:
Йоана Фридрих Михайлова
GitHub: https://github.com/yoanamihaylova

От:
ПМГ „Акад. Боян Петканчин“,
гр. Хасково

Дата: 09.07.2020г.

ПРОЕКТ:

„Таймер Аларма“

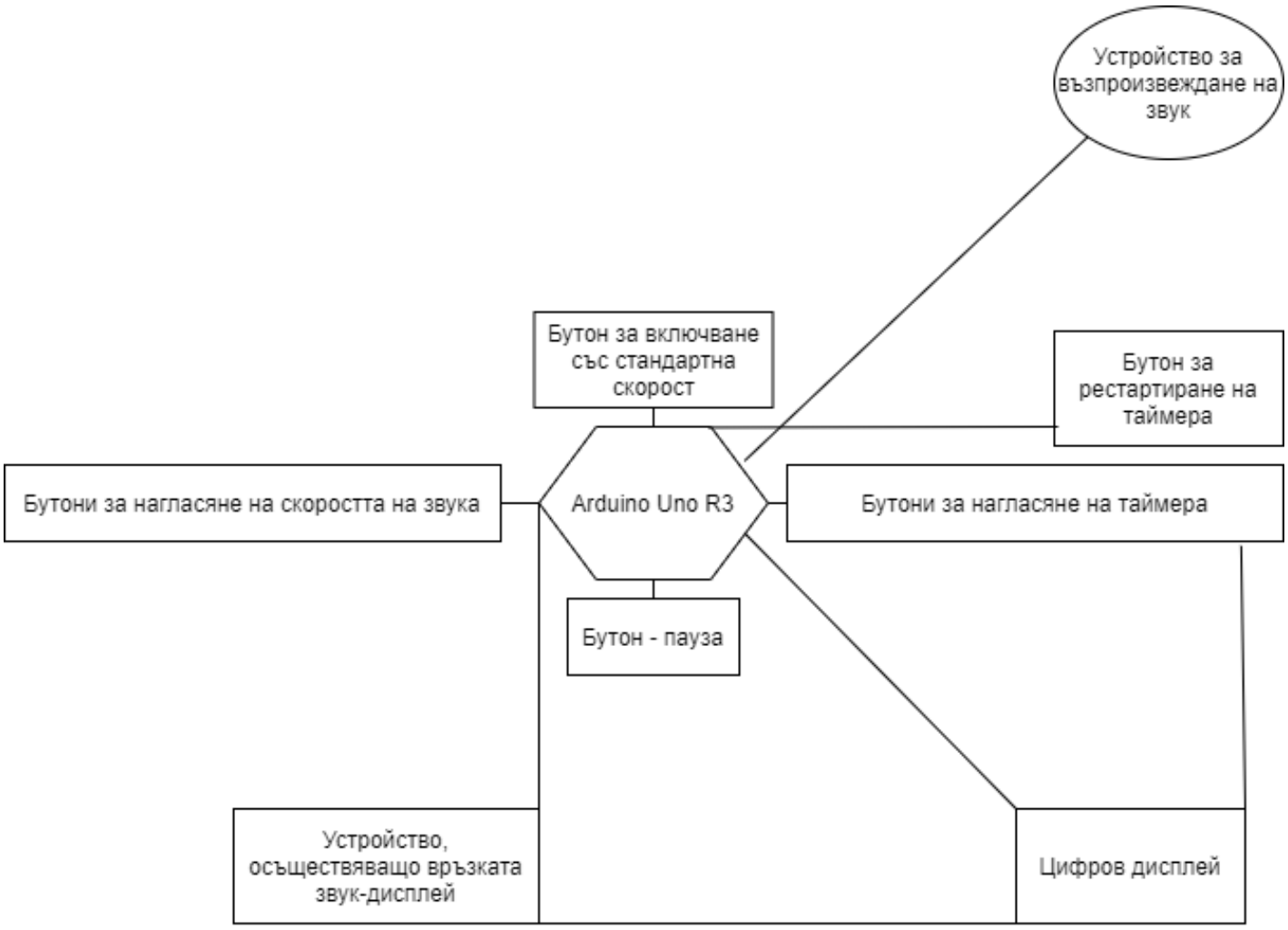
Описание на проекта:

Написана програма на “Tinkercad” за таймер с аларма:

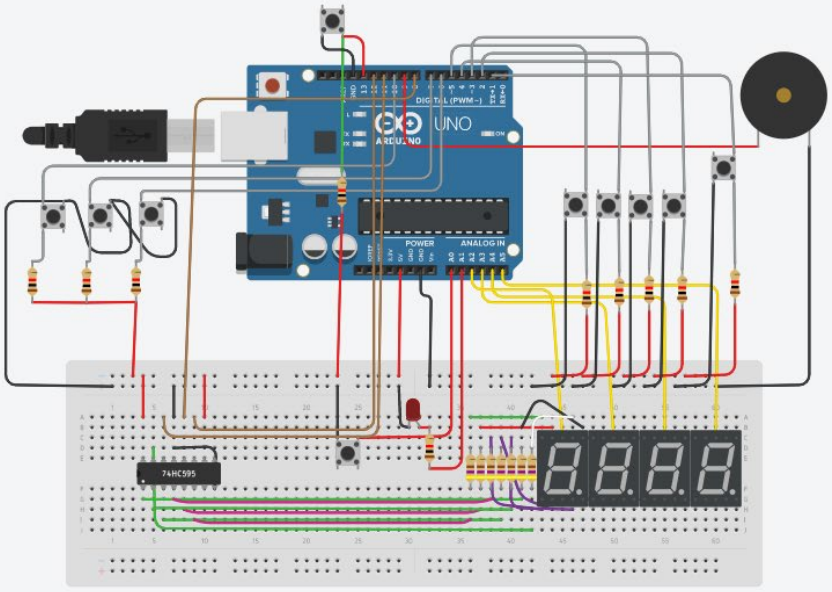
Възпроизвежда се звук за зададеното време на таймера, докато то изтече. Чрез натискане на бутони можем да зададем съответното време, което искаме таймерът да отброява. Трябва да стартираме като можем да определим скоростта на звуците, които чуваме на x1 (стандартна), x2, x3 или x4. Имаме бутон за пауза, както и един за рестартиране на таймера. При стартиране звука се възпроизвежда, а когато времето изтече, звука приключва.

Функционалността на системата се изпълнява от Arduino модула, в който има зададени команди и инструкции под формата на програмен код. Ходът на таймера може да се види от потребителя чрез цифров дисплей, който показва оставащото време в което звука бива чуван.

Блокова схема



Електрическа схема



Списък със съставни части

Name	Quantity	Component
U1	1	Arduino Uno R3
Digit3 Digit4 Digit5 Digit6	4	Anode 7 Segment Display
U2	1	8-Bit Shift Register
R1 R2 R3 R4 R5 R6 R7	7	470 Ω Resistor
PIEZOpiezo_pin	1	Piezo
R8 R19 R20 R21 R13 R14 R15 R16 R17 R18	10	1 kΩ Resistor
Sstart_pin Sspeed4x_pin Sspeed3x_pin Sspeed2x_pin S5 Sreset_btn1 Sseconds_tens Sseconds_units Sminutes_tens Sminutes_units	10	Pushbutton
Dfreeze_led_pin	1	Red LED

Сорс код и описание на функционалността

Описание на функционалността:

Кодът отчита действията на потребителя и изпълнява операциите, зададени от него. Също така, поддържа обновлението на таймера, показан на дисплея, който ни показва оставащото време. Действията на потребителя са: промяна на времето в минути и секунди, пауза, рестартиране, стартиране и задаване на скорост.

Програмен код:

```
#if I2C_DISPLAY
#include <Wire.h>
#include <Adafruit_GFX.h>
#include "Adafruit_LEDBackpack.h"
Adafruit_7segment matrix = Adafruit_7segment();
#endif


//Pin numbers:
const int reset_btn_pin = 1;

const int minutes_tens_pin = 5;
const int minutes_units_pin = 4;
const int seconds_tens_pin = 3;
const int seconds_units_pin = 2;


const int speed2x_pin = 6;
const int speed3x_pin = 7;
const int piezo_pin = 9;
const int speed4x_pin = 10;
const int freeze_btn_pin = A0; //freeze button
const int freeze_led_pin = A1;
const int start_pin = 13;

const int total_alarms = 5;

#if !I2C_DISPLAY
byte digit[4] = {A5, A4, A3, A2};
int latchPin = 8;//Pin connected to ST_CP of 74HC595
```

```
int clockPin = 12;//Pin connected to SH_CP of 74HC595
int dataPin = 11;////Pin connected to DS of 74HC595
#else
const int alarm_pins[total_alarms] = {8, 12, 11, A2, A3}; //pin numbers for
alarm LED
#endif
const int alarm_at_minute[total_alarms] = {0, 5, 10, 15, 20};  //minutes
remaining at which to turn on alarm pin

static boolean pause = 1;
unsigned long speeder = 1000;
int seconds = 0;
int minutes = 0;
static int counter = 0;


void setup()
{
#if !I2C_DISPLAY
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  for (int i = 0; i < 4; i++)
  {
    pinMode(digit[i], OUTPUT);
  }
#else
  for (int i = 0; i < 7; i++)
  {
    pinMode(alarm_pins[i], OUTPUT);
  }
#endif

  pinMode(reset_btn_pin, INPUT_PULLUP);
  pinMode(start_pin, INPUT_PULLUP);

  pinMode(minutes_tens_pin, INPUT_PULLUP);
  pinMode(minutes_units_pin, INPUT_PULLUP);
  pinMode(seconds_tens_pin, INPUT_PULLUP);
  pinMode(seconds_units_pin, INPUT_PULLUP);

  pinMode(speed2x_pin, INPUT_PULLUP);
  pinMode(speed3x_pin, INPUT_PULLUP);
  pinMode(speed4x_pin, INPUT_PULLUP);
  pinMode(freeze_btn_pin, INPUT_PULLUP);
  pinMode(freeze_led_pin, OUTPUT);
  pinMode(piezo_pin, OUTPUT);


  digitalWrite(piezo_pin, 0);

#if !I2C_DISPLAY
  print_serial("Mode: Shift74HC595 IC\n");
```

```
#else

  matrix.begin(0x70);
  print_serial("Mode: I2C\n");
#endif


  reset_countdown();
  print_serial("Ready\n");
}

static unsigned long last_mills = 0;
boolean drawDots = false;



boolean beep = 0;
unsigned char serial_tick = 0;
int last_alarm = -1;

void loop()
{
  unsigned long now_mills = millis();
  if ((now_mills - last_mills) >= speeder)
  {
    last_mills = now_mills;

    if (!pause)
    {
      drawDots = false;
      seconds--;
      if (seconds < 0)
      {
        if (minutes > 0)
        {
          seconds = 59;
          minutes--;
          if (minutes < 0)
          {
            minutes = 0;
          }
        }
        else
        {
          countdown_reached();
        }
      }

      beep = !beep;
      if (beep)
        analogWrite(piezo_pin, 10);
    }
#if ENABLE_SERIAL
```

```
    //show clock on serial monitor every 2 seconds
    serial_tick++;
    if (serial_tick >= 2)
    {
      serial_tick = 0;
      Serial.begin(9600);
      Serial.print(minutes); Serial.print(':'); Serial.println(seconds);
      Serial.end();
    }
#endif

  }
  else if (now_mills > (last_mills + (speeder / 2)))
  {
    drawDots = true;
    digitalWrite(piezo_pin, 0);
  }

  counter = (minutes * 100) + seconds;

#if I2C_DISPLAY
  matrix.writeDigitNum(0, (counter / 1000), drawDots);
  matrix.writeDigitNum(1, (counter / 100) % 10, drawDots);
  matrix.drawColon(drawDots);
  matrix.writeDigitNum(3, (counter / 10) % 10, drawDots);
  matrix.writeDigitNum(4, counter % 10, drawDots);
  matrix.writeDisplay();
  delay(10);
#else
  displayRefresh(counter);
#endif

  check_pins();
}




void countdown_reached()
{
  seconds = 0;
  minutes = 0;
  pause = true;
#if I2C_DISPLAY
  for (int i = 0; i < total_alarms; i++)
  {
    digitalWrite(alarm_pins[i], LOW);
  }
  digitalWrite(alarm_pins[0], HIGH);
#endif
  print_serial("Countdown Reached\n");
}
```

```
void reset_countdown()
{
  print_serial("Countdown reset\n");
  seconds = 0;
  minutes = 0;
  pause = true;
  speeder = 1000;
  last_alarm = -1;
#if I2C_DISPLAY
  for (int i = 0; i < total_alarms; i++)
  {
    digitalWrite(alarm_pins[i], LOW);
  }
#endif
  digitalWrite(freeze_led_pin, 0);
}

void start_countdown()
{
  print_serial("Countdown started\n");
  pause = false;
  speeder = 1000;
  last_alarm = -1;
#if I2C_DISPLAY
  for (int i = 0; i < total_alarms; i++)
  {
    digitalWrite(alarm_pins[i], LOW);
  }
#endif
  digitalWrite(freeze_led_pin, 0);
}
```

```
void check_pins()
{
  if (read_pin(reset_btn_pin))
  {
    reset_countdown();
  }
  if (read_pin(minutes_tens_pin))
  {
    minutes += 10;
  }
  if (read_pin(minutes_units_pin))
  {
    minutes++;
  }
  if (read_pin(seconds_tens_pin))
  {
```

```
    seconds += 10;
  }
  if (read_pin(seconds_units_pin))
  {
    seconds++;
  }
  if (read_pin(start_pin))
  {
    start_countdown();
  }
  if (read_pin(speed2x_pin))
  {
    speeder = 500;
    print_serial("Speed 2x\n");
  }
  if (read_pin(speed3x_pin))
  {
    speeder = 250;
    print_serial("Speed 3x\n");
  }
  if (read_pin(speed4x_pin))
  {
    speeder = 125;
    print_serial("Speed 4x\n");
  }
  if (read_pin(freeze_btn_pin))
  {
    pause = true;
    digitalWrite(freeze_led_pin, 1);
    print_serial("Freeze\n");
  }


  while (seconds >= 60)
  {
    seconds = seconds - 60;
    minutes++;
    if (minutes > 99)
    {
      minutes = 0;
      seconds = 0;
    }
  }

  if (!pause)
  {
#if I2C_DISPLAY
    for (int i = 0; i < total_alarms; i++)
    {
      digitalWrite(alarm_pins[i], LOW);
    }
#endif
    for (int i = 0; i < 7; i++)
```

```
    {
      if ((minutes == alarm_at_minute[i]) && ((seconds == 0)))
      {
        if(last_alarm != i)
        {
        print_serial("Alarm\t" + String(alarm_at_minute[i]) + " mins\n");
        last_alarm = i;
        }
#if I2C_DISPLAY
        digitalWrite(alarm_pins[i], HIGH);
#endif
        break;
      }
    }
  }

}

int last_btn = -1;

boolean read_pin(int pin_no)
{
  if (last_btn != -1)
  {
    if ((digitalRead(last_btn) == LOW) && (digitalRead(last_btn) == LOW))
    {
      return false;
    }
    else
      last_btn = -1;
  }

  if ((digitalRead(pin_no) == LOW) && (digitalRead(pin_no) == LOW))
  {
    unsigned long count_press = 0;
    while (count_press < 10)
    {
      if (digitalRead(pin_no) == HIGH)
      {
        return false;
      }
      count_press++;
      delay(1);
    }
    print_serial("Pin " + String(pin_no) + " LOW\n");
    last_btn = pin_no;
    return true;
  }
  return false;
}

void print_serial(String toprint)
{
```

```
#if ENABLE_SERIAL
  Serial.begin(9600);
  Serial.print(toprint);
  Serial.end();
#endif
}



#if !I2C_DISPLAY
int last_digit = 0;
void displayRefresh(int count)
{
  last_digit++;
  if (last_digit >= 4)
    last_digit = 0;
  for (int i = 0; i < 4; i++)
  {
    if (i == last_digit)
    {
      displayDigit(extractDigit(count, i + 1));
      digitalWrite(digit[i], HIGH);
    }
    else
    {
      digitalWrite(digit[i], LOW);
    }
  }
  delay(5);
}
void displayDigit(int d)
{
  char number[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,
0x6F};
  digitalWrite(latchPin, LOW);
  // shift out the bits:
  shiftOut(dataPin, clockPin, MSBFIRST, ~number[d]);

  //take the latch pin high so the LEDs will light up:
  digitalWrite(latchPin, HIGH);
}

int extractDigit(int V, int P)
{
  return int(V / (pow(10, P - 1))) % 10;
}
#endif
```

Източници : https://www.tinkercad.com/things/bntqq7IEVd2-arduino-countdown-timer

https://www.robotshop.com/community/forum/t/arduino-101-timers-and-interrupts/13072