**Изготвила:**

Йоана Фридрих Михайлова

**GitHub:**

[https://github.com/yoanamihaylova](https://github.com/yoanamihaylova/ItKariera/tree/master/II/IntroducingToOperatingSystemsAndSystemIntegration)

**TinkerCad:**

<https://www.tinkercad.com/things/g4D0KSBzaZh-timer-alarm/editel?sharecode=1AeEAdWtEfvWXCm5hpA1iFLPe2t76Z-GbS5Z5O5nh2c>

**Дата:** 09.07.2020г**.**

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

НП „ИТ КАРИЕРА“  
Модул VIII   
Разработка на софтуер

**“Таймер Аларма”**

Проект:

**„Таймер Аларма“**

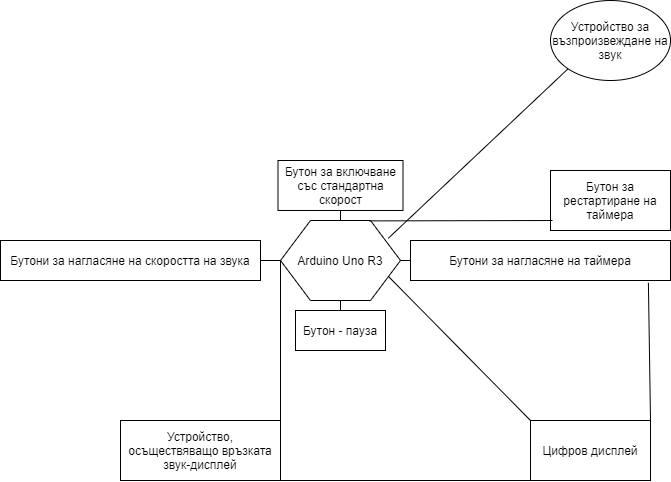
**Описание на проекта:**

Написана програма на “Tinkercad” за таймер с аларма:

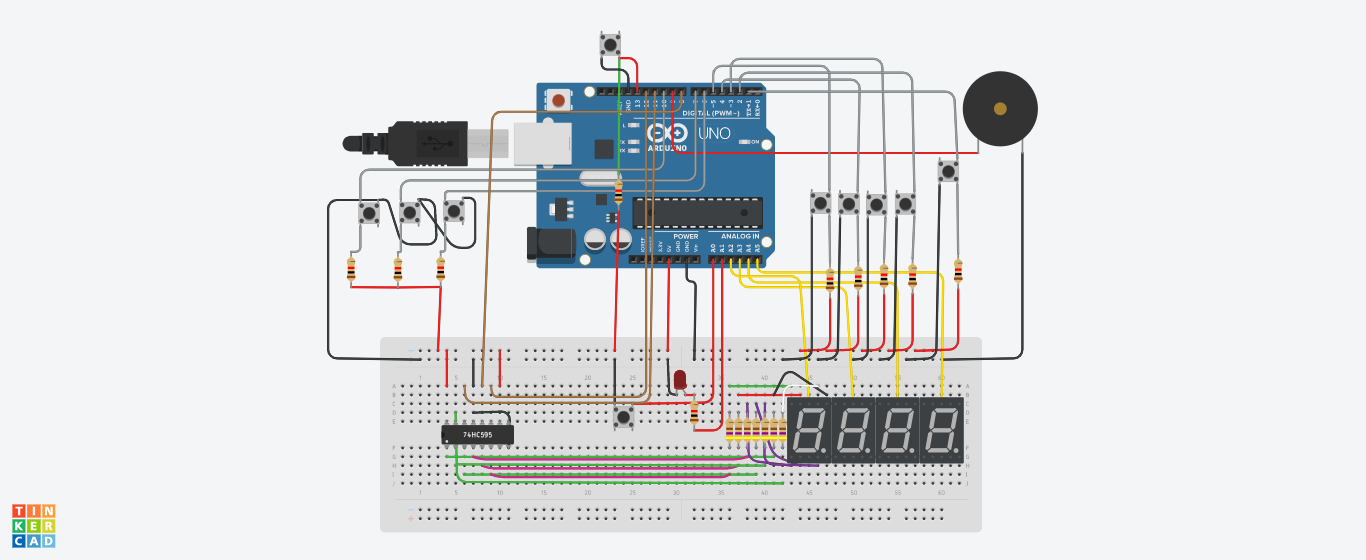
Възпроизвежда се звук за зададеното време на таймера, докато то изтече. Чрез натискане на бутони можем да зададем съответното време, което искаме таймерът да отброява. Трябва да стартираме като можем да определим скоростта на звуците, които чуваме на х1 (стандартна), х2, х3 или х4. Имаме бутон за пауза, 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.

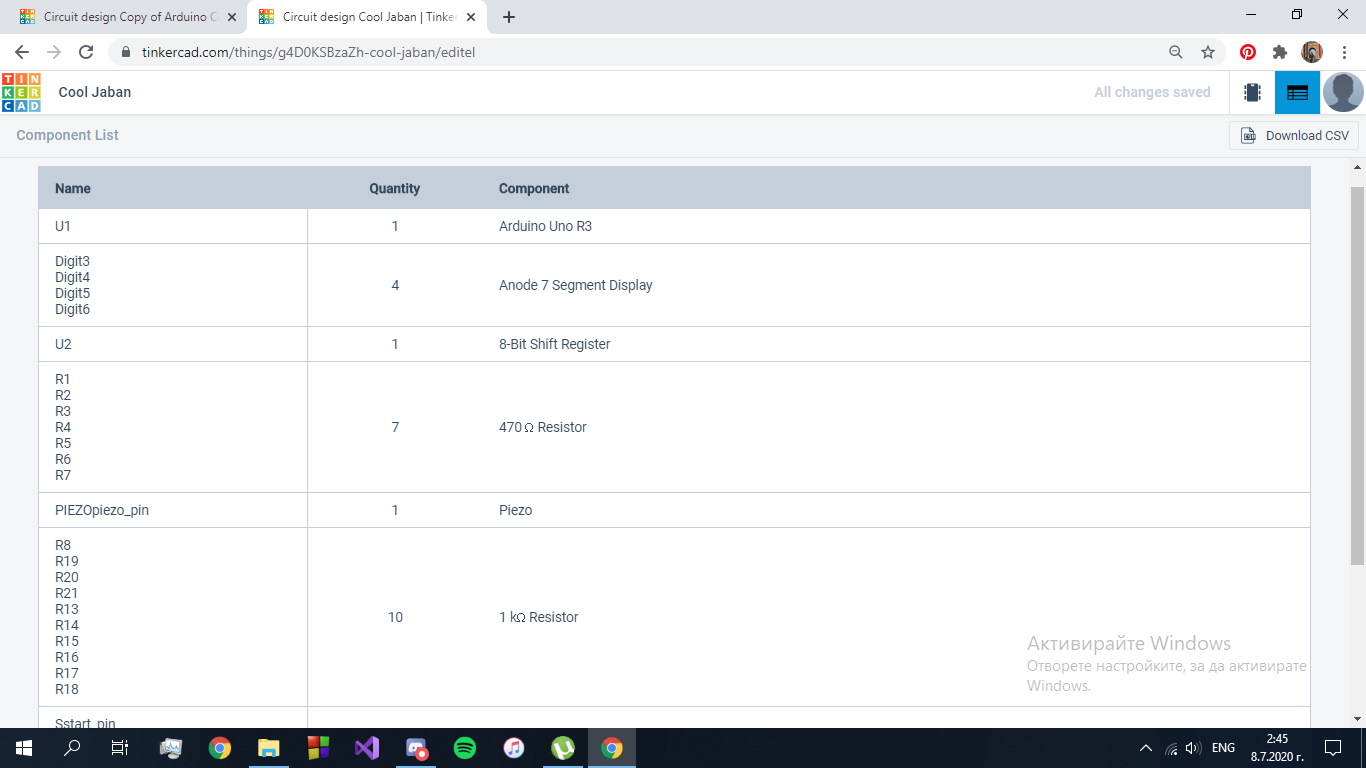
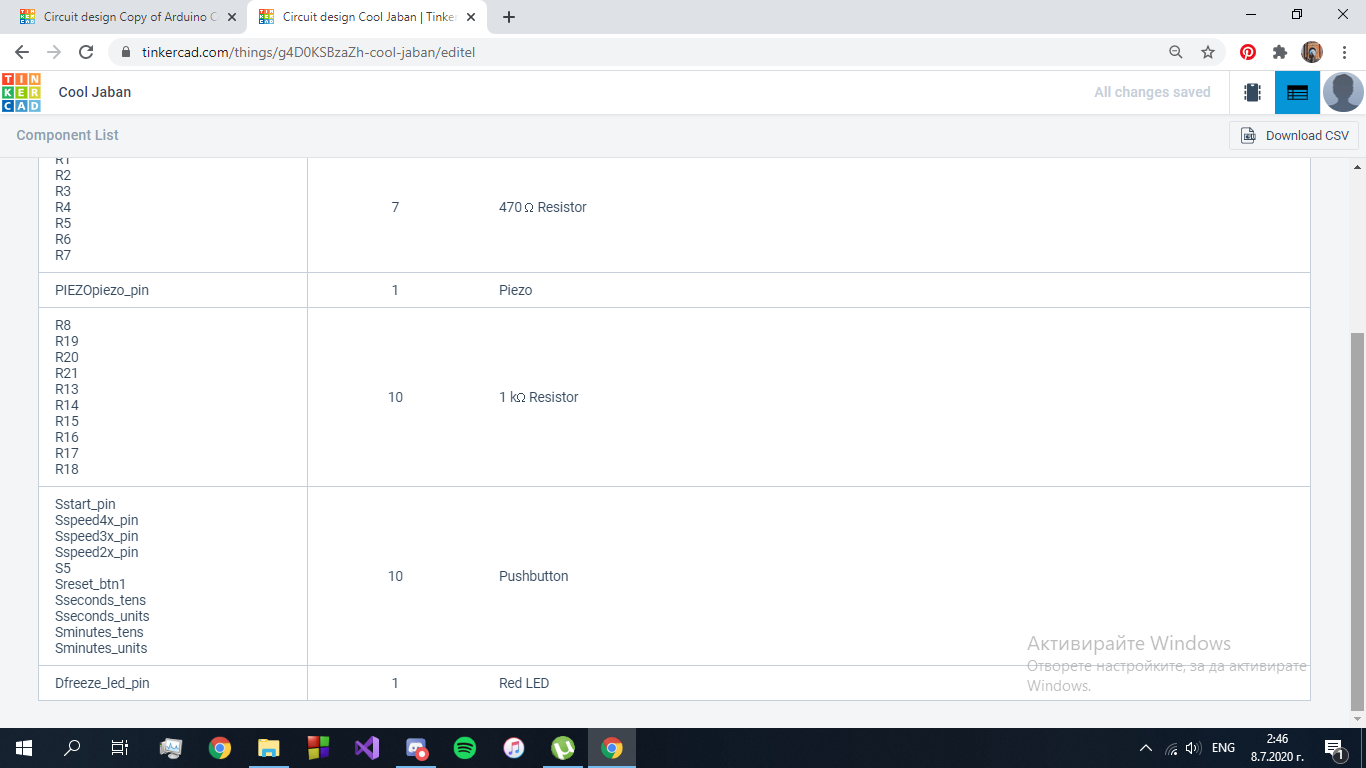
**Блокова схема**



**Електрическа схема**



**Списък със съставни части**



**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.

**Пограмен 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>

<https://www.smartdraw.com/block-diagram/block-diagram-maker.htm>