Αναφορά 2ης Εργαστηριακής Άσκησης

Στοιχεία Ομάδας:

Δημήτριος Κωστορρίζος, up1054419@upnet.gr

Λάμπρος Παπαδόπουλος, up1054433@upnet.gr

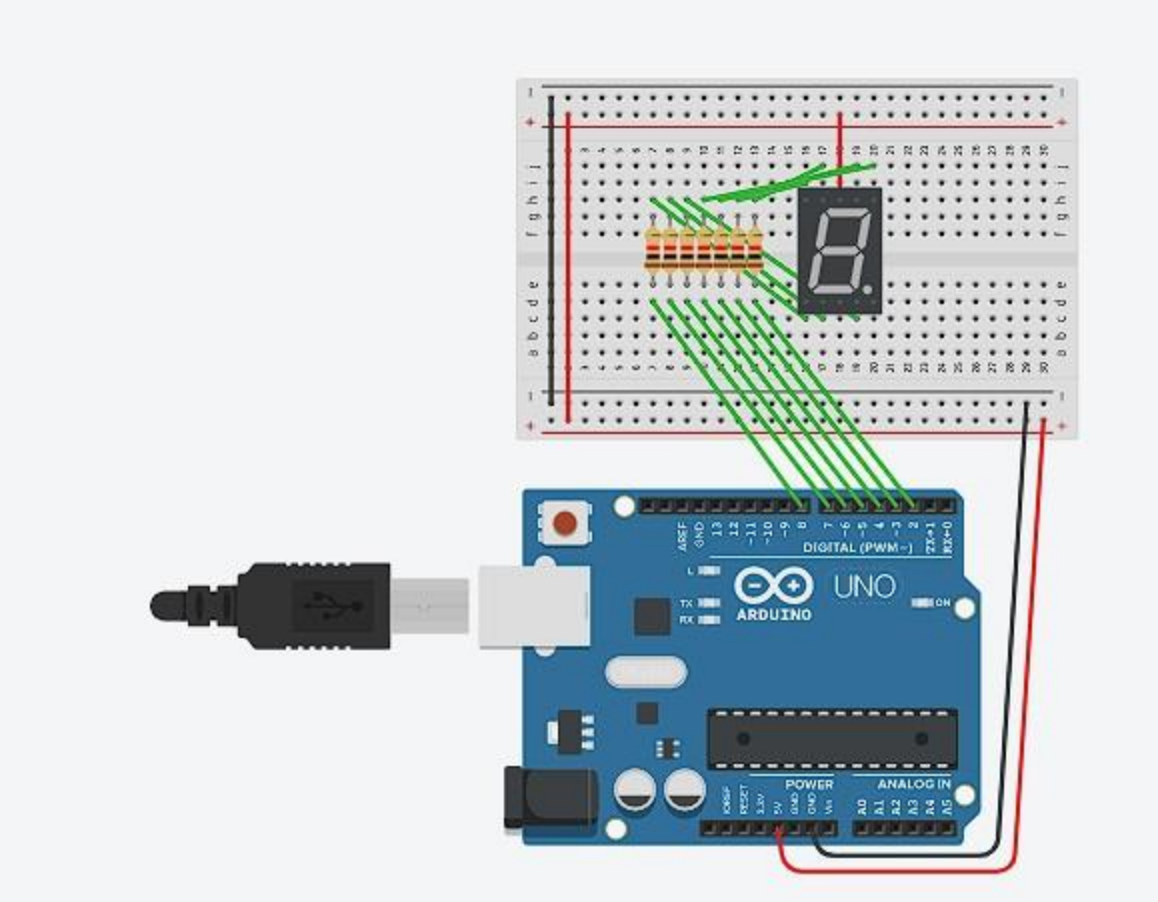
Ομάδα Β

Τμήμα: 1

1ο Υποερώτημα

Tinkercad Link: https://www.tinkercad.com/things/6ivzXWow1qS-askisi2/editel

Το 1ο υποερώτημα, υλοποιήθηκε με βάση την συνδεσμολογία, που φαίνεται στο παρακάτω σχήμα.



Οι περιοχές, στις οποίες η διαβάθμιση της φωτεινότητας γίνεται, είναι

- 12 – 21
- 43 – 48
- 67 – 74
- 91 – 96

Στο Arduino, οι χαρακτήρες δίνονται από το Serial Command Line, ενώ στον AT91 η ίδια είσοδο δίνεται από το πληκτρολόγιο.

Κώδικες

Arduino

```
void setup()
{
	Serial.begin(9600);
	for(int n = 2; n <= 8; n++)
  {
	pinMode(n, OUTPUT);
		digitalWrite(n, LOW);
  }
}

int next = 100;
void loop()
{
	char tmp;
	if (Serial.available())
	{
		tmp = Serial.read();

		if (tmp == 'd')
		{
			if(next >= 0)
			{
			next-=20;
			}
		}

		if (tmp == 'u')
		{
			if(next <= 100)
			{
				next+=20;
			}
		}
	}
	else
	{
		for(int index = 0; index < 100; index++)
		{
			if(index < next)
			{
				for(int n = 2; n <= 8; n++)
```

```
                                        {
                                          digitalWrite(n, LOW);
                                        }
                                }
                                else
                                {
                                        for(int n = 2; n <= 8; n++)
                                        {
                                         digitalWrite(n, HIGH);
                                        }
                                }
                        }
                }
}
```

AT91

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <stdio.h>
#include <stdlib.h>
#include <header.h>

unsigned int buttonState = BUT_IDLE;//RESET btn

PIO* pioa = NULL;
unsigned int periodCounter = 0;
unsigned int next = 100;
unsigned int countingFlag = 0;


int main(int argc, const char * argv[])
{
        int currentNext = 0;
        unsigned int gen;

        STARTUP;
        tc->Channel_0.RC = 100;//Frequency =100 pulses
        tc->Channel_0.CMR = 2084;
```

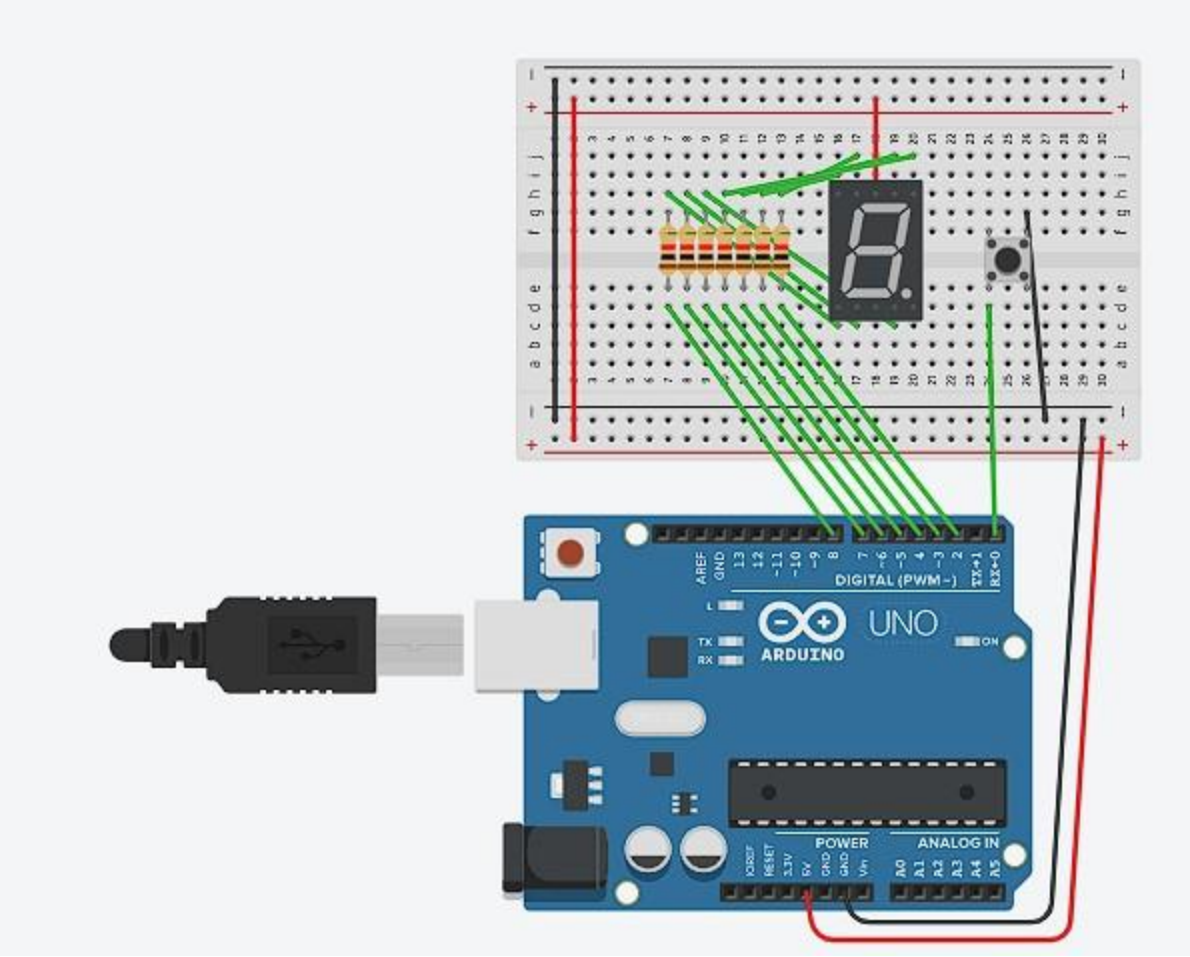
```
	tc->Channel_0.IDR = 0xFF;
	tc->Channel_0.IER = 0x10;
	aic->FFER = (1<<PIOA_ID) | (1<<TC0_ID);
	aic->IECR = (1<<PIOA_ID) | (1<<TC0_ID);

	pioa->PUER = 0x200; //PULLUP Inputs
	pioa->ODR = 0x200;//Bit 9 input

	pioa->OER = 0xFF; //Bits 0-6 Outpouts
	pioa->SODR = 0xFF;  //Turn on Display

	gen = pioa->ISR;//Interupts cleanup
	pioa->PER = 027F; //0-6,9 generic
	gen = tc->Channel_0.SR;//Interupts cleanup
	aic->ICCR = (1<<PIOA_ID)|(1<<TC0_ID);//Interupts cleanup
	pioa->IER = 0x91;//Interupt on 9

	tc->Channel_0.CCR = 0x02; //Timer stopped

	// Read characters from the keyboard
	while((tmp = getchar()) != 'e')
	{
		// Reset current next to 0
		if(currentNext > 100)
			currentNext = 0;

		// If current next is less that next
		if(currentNext < next)
			// Turn off the 7-segment display
			pioa->SODR = 0xFF;
		else
			// Turn on the 7-segment display
			pioa->CODR = 0xFF;

		// Increase the current next
		currentNext++;
	}
}

void FIQ_handler(void)
{
	if(fiq & (1<<PIOA_ID))
	{// Διακοπές από την παράλληλη
		data_in = pioa->ISR;//Interupt cleanup
```

```
		aic->ICCR = (1<<PIOA_ID);//Interupts cleanup
		data_in = pioa->PDSR;//Data inputs read

		if(!(data_in & 0x200))
		{
		 if(buttonState == BUT_IDLE)
		 {
			buttonState = BUT_PRESSED;
			if(tc->Channel_0.CCR == 0x05)
			{
				tc->Channel_0.CCR = 0x02;
				periodCounter = 0;
			}
			else
			{
				tc->Channel_0.CCR = 0x05;
			}
		  }
		  else
		  {
			if(buttonState == BUT_PRESSED)
			{
					buttonState = BUT_IDLE;
			}
		  }
		}
}


if( fiq & (1<<TC0_ID) ){//Timer Interupts
		data_out = tc->Channel_0.SR;//Interupts cleanup
		aic->ICCR = (1<<TC0_ID);//Interupts cleanup
		data_out = pioa->ODSR;//Data exits read

		periodCounter++;

		// If 20 periods have passed
		if(periodCounter == 20)
		{
			// Reset the period counter
			periodCounter = 0;
			if(next == 100)
			{
				// Set the flag to decrease the next
```

```
                countingFlag = 0;
            }
            else
            {
                if(next == 0)
                {
                    // Set the flag to increase the next
                    countingFlag = 1;
                }
            }
            if(countingFlag)
            {
                // Increase the next
                next++;
            }
            else
            {
                // Decrease the next
                next--;
            }
        }

        // Set the channel ccr to 0x05
        tc->Channel_0.CCR = 0x05;
    }
}
```

2ο Υποερώτημα

Το 2ο υποερώτημα, υλοποιήθηκε με βάση την συνδεσμολογία, που φαίνεται στο παρακάτω σχήμα.



Η διαδικασία παλινδρόμησης μεταξύ της μέγιστης φωτεινότητας και της ελάχιστης, ξεκινά με το πάτημα του κουμπιού και σταματά με το επόμενο πάτημα. Άρα, όταν το πάτημα του κουμπιού αντιστοιχεί σε περιττό αριθμό η παλινδρόμηση ξεκινά και σταματά όταν το πάτημα του κουμπιού αντιστοιχεί σε άρτιο αριθμό. Λόγω της ταχύτητας εκτέλεσης των εντολών και του τρόπου λειτουργίας του 7-segment display, είναι αρκετά δύσκολο να παρατηρηθεί τόσο η κατάσταση της μέγιστης φωτεινότητας, όσο και η κατάσταση ελάχιστης φωτεινότητας.

Κώδικες

Arduino

```
//AT91 emulation code by Theodoros Simopoulos
```

```
#define HIGH 0x1
#define LOW 0x0

//AT91 special
#define BUT_IDLE 0
#define BUT_PRESSED 1
#define BUT_RELEASED 2
#define LED_IDLE 0
#define LED_FLASHING 1

//Arduino special
#define continue_count  0x01
#define stop_count  0x02
#define reset_and_start_count  0x05

unsigned int PIOA_int, TC_int; //ομοίως int με πιθανές τιμές 0|1
unsigned long clk_pulse_counter;
unsigned int Channel_0_CCR;
unsigned int previous_button_state;
unsigned long Channel_0_RC;

//Μεταβλητές και σταθερές εργαστηριακής άσκησης
unsigned int button_state = BUT_IDLE;
unsigned int led_state = LED_IDLE;
int next =100;
int flag = 0;
int status = 0;

void setup()
{
  pinMode(1, OUTPUT);          //ODR, PUER
        pinMode(2, OUTPUT);    //ODR, PUER
  pinMode(3, OUTPUT);          //ODR, PUER
  pinMode(4, OUTPUT);          //ODR, PUER
  pinMode(5, OUTPUT);          //ODR, PUER
  pinMode(6, OUTPUT);          //ODR, PUER
  pinMode(7, OUTPUT);          //ODR, PUER
        digitalWrite(1, LOW);
        digitalWrite(2, LOW);
        digitalWrite(3, LOW);
        digitalWrite(4, LOW);
        digitalWrite(5, LOW);
        digitalWrite(6, LOW);
        digitalWrite(7, LOW);
```

```
        pinMode(0, INPUT_PULLUP);
        clk_pulse_counter=0;
        Channel_0_CCR = stop_count;
        previous_button_state = BUT_PRESSED;
        Channel_0_RC = 100;
        PIOA_int=LOW;
        TC_int=LOW;
}

//main
void loop()
{
 //Ενεργοποίηση διακοπής από τον timer/counter
 if (Channel_0_CCR == reset_and_start_count)
 {
  clk_pulse_counter = 0;
  Channel_0_CCR = continue_count;
  }

 if (Channel_0_CCR == continue_count)
 {
  clk_pulse_counter = clk_pulse_counter+1;
  if (clk_pulse_counter == Channel_0_RC)
   {
        clk_pulse_counter =0;
        TC_int = HIGH;
   }
 }


 //Ενεργοποίηση διακοπής από την παράλληλη A (PIOA)
 button_state = digitalRead(0);
 if (button_state != previous_button_state)
 {
  previous_button_state = button_state;
  PIOA_int = HIGH;
 }

 if (PIOA_int | TC_int) //aic->ICCR
 {
  FIQ_handler();
        PIOA_int = LOW;
 }
 if(flag == 1)
```

```
{
        // Set next after every 100 clock pulse
 if(status == 0 && (clk_pulse_counter % 100) == 0)
        {
                next-=1;
        }
 if(status == 1 && (clk_pulse_counter % 100) == 0)
 {
                next+=1;
        }

        // Set the status to increase the next
        if(next == 0)
        {
                status = 1;
        }

        // Set the status to decrease the next
        if(next == 100)
        {
                status = 0;
        }

        // Set the 7-segment display
        if(clk_pulse_counter < next)
        {
                digitalWrite(1, LOW);
                digitalWrite(2, LOW);
                digitalWrite(3, LOW);
                digitalWrite(4, LOW);
                digitalWrite(5, LOW);
                digitalWrite(6, LOW);
                digitalWrite(7, LOW);
        }
        else
        {
                digitalWrite(1, HIGH);
                digitalWrite(2, HIGH);
                digitalWrite(3, HIGH);
                digitalWrite(4, HIGH);
                digitalWrite(5, HIGH);
                digitalWrite(6, HIGH);
                digitalWrite(7, HIGH);
        }
```

```
 }
}

void FIQ_handler(void)
{
 if (PIOA_int == HIGH)// Η διακοπή προκλήθηκε από την PIOA
 {
  PIOA_int = LOW;
  if(button_state == BUT_IDLE){
   button_state = BUT_PRESSED;
   if( led_state == LED_IDLE ){
    Channel_0_CCR = 0x05;
    led_state = LED_FLASHING;
   }
   else{
    Channel_0_CCR = 0x02;
    led_state = LED_IDLE;
    }
   }
  else
        {
   if(button_state == BUT_PRESSED)
    button_state = BUT_IDLE;
  }
 }

 if (TC_int == HIGH)// Η διακοπή προκλήθηκε από τον timer/counter
 {
  TC_int = LOW;
  PORTD = PORTD ^ 0x02;         //pioa->SODR = data_out | 0x02
                                                        //pioa->CODR = data_out & 0x02
 }
 // Set the flag to count descending or ascending
        if(flag == 1)
        {
                flag = 0;
        }
        else
        {
                flag = 1;
        }
}
```

AT91

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <stdio.h>
#include <stdlib.h>
#include <header.h>

#define BUT_IDLE 0
#define BUT_PRESSED 1

PIO* pioa = NULL;
AIC* aic = NULL;
TC* tc = NULL;

unsigned int periodCounter = 0;
unsigned int next = 100;
unsigned int status = 0;
unsigned int buttonState = BUT_IDLE;

int main( int argc, const char * argv[] )
{
```

```
int currentNext = 0;
unsigned int gen;

STARTUP;
tc->Channel_0.RC = 100;
tc->Channel_0.CMR = 2084;
tc->Channel_0.IDR = 0xFF;
tc->Channel_0.IER = 0x10;
aic->FFER = (1<<PIOA_ID) | (1<<TC0_ID);
aic->IECR = (1<<PIOA_ID) | (1<<TC0_ID);

pioa->PUER = 0x200;
pioa->ODR = 0x200;

pioa->OER = 0xFF;
pioa->SODR = 0xFF;

gen = pioa->ISR;
pioa->PER = 0x27F;
gen = tc->Channel_0.SR;
aic->ICCR = (1<<PIOA_ID)|(1<<TC0_ID);
pioa->IER = 0x200;

tc->Channel_0.CCR = 0x02;

// Read until the 'e' character is inserted
while((tmp=getchar()) != 'e')
{
        // Reset the current next counter
```

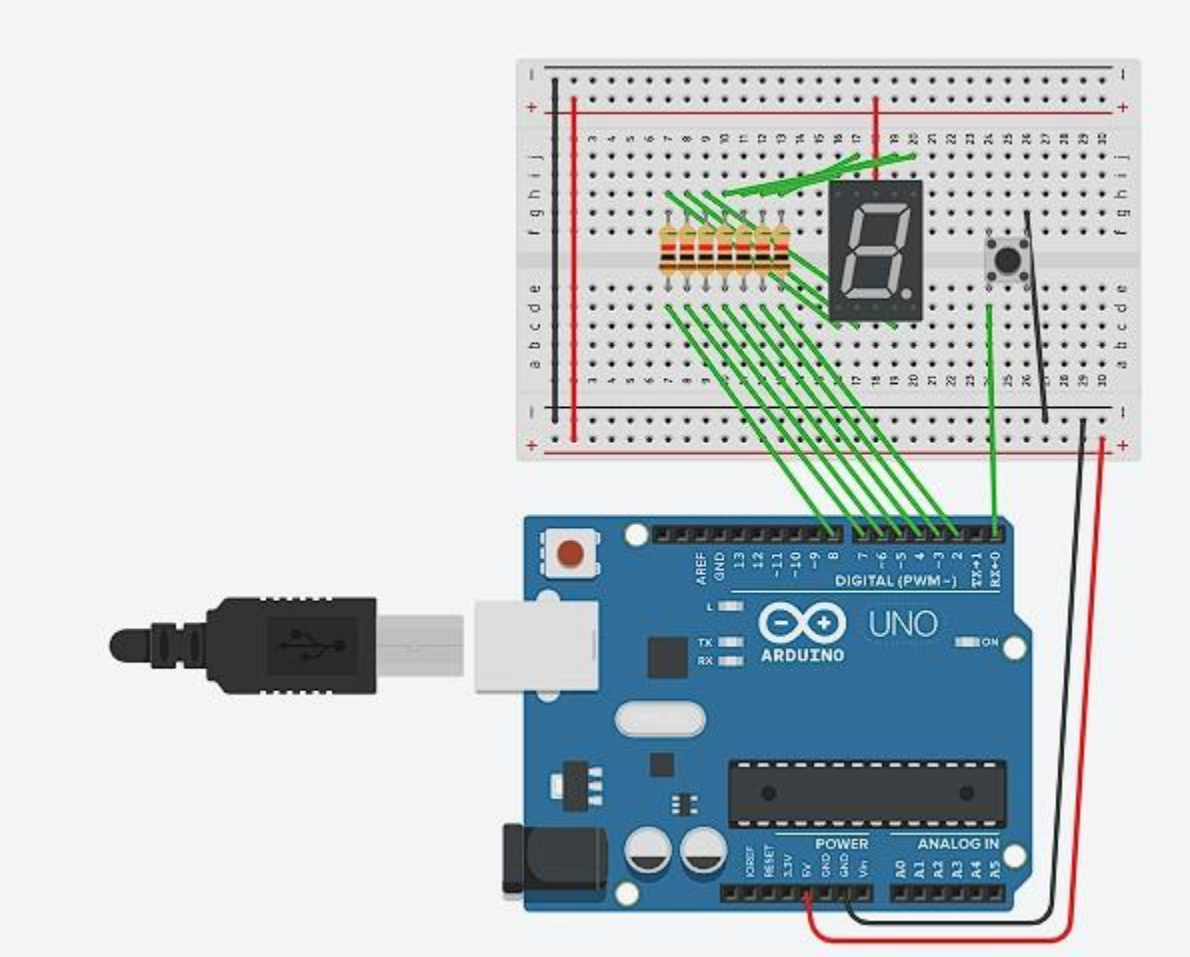
```
                if(currentNext > 100)
                        currentNext = 0;

                if(currentNext < next)
                        // Turn off the 7-segment display
                        pioa->SODR = 0xFF;
                else
                        // Turn on the 7-segment display
                        pioa->CODR = 0xFF;

                // Increase the current next
                currentNext++;
        }


}

void FIQ_handler(void)
{

        if(fiq & (1<<PIOA_ID))
        {
                data_in = pioa->ISR;
                aic->ICCR = (1<<PIOA_ID);
                data_in = pioa->PDSR;

                if(!( data_in & 0x200 ))
                {
                        if(buttonState == BUT_IDLE)
```

```
                {
                        buttonState = BUT_PRESSED;
                        if(tc->Channel_0.CCR == 0x05)
                        {
                                tc->Channel_0.CCR = 0x02;
                                periodCounter = 0;
                        }
                        else
                        {
                                tc->Channel_0.CCR = 0x05;
                        }

                }
                else
                  if(buttonState == BUT_PRESSED)
                                buttonState = BUT_IDLE;
        }
}



if(fiq & (1<<TC0_ID))
{
        data_out = tc->Channel_0.SR;
        aic->ICCR = (1<<TC0_ID);
        data_out = pioa->ODSR;

        periodCounter++;

        // If 20 periods have passed
```

```
            if(periodCounter == 20)
            {
                // Reset the period counter
                periodCounter = 0;

                // Set the flag to count descending
                if(next == 100)
                    status = 0;
                else
                {
                    // Set the flag to count ascending
                    if(next == 0)
                        status = 1;
                }

                // Set the next
                if(status)
                    next++;
                else
                    next--;
            }

            // Reset the clock counter
            tc->Channel_0.CCR = 0x05;
        }
}
```

3ο Υποερώτημα

Το 3ο υποερώτημα, υλοποιήθηκε με βάση την συνδεσμολογία, που φαίνεται στο παρακάτω σχήμα.



Η διαδικασία παλινδρόμησης μεταξύ της μέγιστης φωτεινότητας και της ελάχιστης, ξεκινά με το πάτημα του κουμπιού και σταματά με το επόμενο πάτημα. Άρα, όταν το πάτημα του κουμπιού αντιστοιχεί σε περιττό αριθμό η παλινδρόμηση ξεκινά και σταματά όταν το πάτημα του κουμπιού αντιστοιχεί σε άρτιο αριθμό. Λόγω της ταχύτητας εκτέλεσης των εντολών και του τρόπου λειτουργίας του 7-segment display, είναι αρκετά δύσκολο να παρατηρηθεί τόσο η κατάσταση της μέγιστης φωτεινότητας, όσο και η κατάσταση ελάχιστης φωτεινότητας. Τα μηνύματα εμφάνισης είναι τοποθετημένα στην main συνάρτηση και όχι στην συνάρτηση διακοπών.

Κώδικες

Arduino

```
//AT91 emulation code by Theodoros Simopoulos
#define HIGH 0x1
#define LOW 0x0

//AT91 special
#define BUT_IDLE 0
#define BUT_PRESSED 1
#define BUT_RELEASED 2
#define LED_IDLE 0
#define LED_FLASHING 1

//Arduino special
#define continue_count  0x01
#define stop_count  0x02
#define reset_and_start_count  0x05

unsigned int PIOA_int, TC_int; //ομοίως int με πιθανές τιμές 0|1
unsigned long clk_pulse_counter;
unsigned int Channel_0_CCR;
unsigned int previous_button_state;
unsigned long Channel_0_RC;

//Μεταβλητές και σταθερές εργαστηριακής άσκησης
unsigned int button_state = BUT_IDLE;
unsigned int led_state = LED_IDLE;
int next =100;
int flag = 0;
int status = 0;

void setup()
{
   pinMode(1, OUTPUT);          //ODR, PUER
         pinMode(2, OUTPUT);    //ODR, PUER
   pinMode(3, OUTPUT);          //ODR, PUER
   pinMode(4, OUTPUT);          //ODR, PUER
   pinMode(5, OUTPUT);          //ODR, PUER
   pinMode(6, OUTPUT);          //ODR, PUER
   pinMode(7, OUTPUT);          //ODR, PUER
         digitalWrite(1, LOW);
```

```
        digitalWrite(2, LOW);
        digitalWrite(3, LOW);
        digitalWrite(4, LOW);
        digitalWrite(5, LOW);
        digitalWrite(6, LOW);
        digitalWrite(7, LOW);
        pinMode(0, INPUT_PULLUP);
        clk_pulse_counter=0;
        Channel_0_CCR = stop_count;
        previous_button_state = BUT_PRESSED;
        Channel_0_RC = 100;
        PIOA_int=LOW;
        TC_int=LOW;
}

//main
void loop()
{
 //Ενεργοποίηση διακοπής από τον timer/counter
 if (Channel_0_CCR == reset_and_start_count)
 {
  clk_pulse_counter = 0;
  Channel_0_CCR = continue_count;
  }

 if (Channel_0_CCR == continue_count)
 {
  clk_pulse_counter = clk_pulse_counter+1;
  if (clk_pulse_counter == Channel_0_RC)
   {
        clk_pulse_counter =0;
        TC_int = HIGH;
   }
 }


 //Ενεργοποίηση διακοπής από την παράλληλη A (PIOA)
 button_state = digitalRead(0);
 if (button_state != previous_button_state)
 {
  previous_button_state = button_state;
  PIOA_int = HIGH;
 }
```

```
if (PIOA_int | TC_int) //aic->ICCR
{
 FIQ_handler();
        PIOA_int = LOW;
}
if(flag == 1)
{
        // Set next after every 100 clock pulse
 if(status == 0 && (clk_pulse_counter % 100) == 0)
        {
                next-=1;
        }
 if(status == 1 && (clk_pulse_counter % 100) == 0)
 {
                next+=1;
        }

        // Set the status to increase the next
        if(next == 0)
        {
                status = 1;
        }

        // Set the status to decrease the next
        if(next == 100)
        {
                status = 0;
        }

        // Set the 7-segment display
        if(clk_pulse_counter < next)
        {
                digitalWrite(1, LOW);
                digitalWrite(2, LOW);
                digitalWrite(3, LOW);
                digitalWrite(4, LOW);
                digitalWrite(5, LOW);
                digitalWrite(6, LOW);
                digitalWrite(7, LOW);
        }
        else
        {
                digitalWrite(1, HIGH);
                digitalWrite(2, HIGH);
```

```
                digitalWrite(3, HIGH);
                digitalWrite(4, HIGH);
                digitalWrite(5, HIGH);
                digitalWrite(6, HIGH);
                digitalWrite(7, HIGH);
        }
 }
}

void FIQ_handler(void)
{
 if (PIOA_int == HIGH)// Η διακοπή προκλήθηκε από την PIOA
 {
  PIOA_int = LOW;
  if(button_state == BUT_IDLE){
   button_state = BUT_PRESSED;
   if( led_state == LED_IDLE ){
    Channel_0_CCR = 0x05;
    led_state = LED_FLASHING;
   }
   else{
    Channel_0_CCR = 0x02;
    led_state = LED_IDLE;
    }
   }
  else
        {
   if(button_state == BUT_PRESSED)
    button_state = BUT_IDLE;
  }
 }

 if (TC_int == HIGH)// Η διακοπή προκλήθηκε από τον timer/counter
 {
  TC_int = LOW;
  PORTD = PORTD ^ 0x02;         //pioa->SODR = data_out | 0x02
                                                        //pioa->CODR = data_out & 0x02
 }
 // Set the flag to count descending or ascending
        if(flag == 1)
        {
                flag = 0;
        }
        else
```

```
                {
                        flag = 1;
                }
}
```

AT91

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <stdio.h>
#include <stdlib.h>
#include <header.h>

#define BUT_IDLE 0
#define BUT_PRESSED 1

PIO* pioa = NULL;
AIC* aic = NULL;
TC* tc = NULL;

unsigned int currentPeriod = 0;
unsigned int next = 100;
unsigned int status = 0;
unsigned int fail = 0;
unsigned int success = 1;
```

```
unsigned int currentNext = 0;
unsigned int periodCounter = 20;
unsigned int fail_message = 0;
unsigned int buttonState = BUT_IDLE;

int main( int argc, const char * argv[] )
{
	unsigned int gen;

	STARTUP;

	tc->Channel_0.RC = 100;
	tc->Channel_0.CMR = 2084;
	tc->Channel_0.IDR = 0xFF;
	tc->Channel_0.IER = 0x10;
	aic->FFER = (1<<PIOA_ID) | (1<<TC0_ID);
	aic->IECR = (1<<PIOA_ID) | (1<<TC0_ID);

	pioa->PUER = 0x200;
	pioa->ODR = 0x200;

	pioa->OER = 0xFF;
	pioa->SODR = 0xFF;

	gen = pioa->ISR;
	pioa->PER = 0x27F;
	gen = tc->Channel_0.SR;
	aic->ICCR = (1<<PIOA_ID)|(1<<TC0_ID);
	pioa->IER = 0x200;
```

```
    tc->Channel_0.CCR = 0x02;

    // If 'e' is pressed, exit
    while((tmp=getchar()) != 'e')
    {
            if(fail_message == 1)
            {

                    // Reset the error message flag
                    fail_message = 0;
                    printf("Error: Failure.\n");
            }

            // Reset the current next counter
            if(currentNext > 100)
  currentNext = 0;

if(currentNext < next)
                    // Turn off the 7-segment display
  pioa->SODR = 0xFF;
else
                    // Turn on the 7-segment display
  pioa->CODR = 0xFF;

            // Increase the current next
currentNext++;
    }
```

```
}

void FIQ_handler(void)
{
	if( fiq & (1<<PIOA_ID) )
	{
		data_in = pioa->ISR;
		aic->ICCR = (1<<PIOA_ID);
		data_in = pioa->PDSR;

		if(!( data_in & 0x200 ))
		{
			if(buttonState == BUT_IDLE)
			{
				buttonState = BUT_PRESSED;
				if(tc->Channel_0.CCR == 0x05 || tc->Channel_0.CCR == 0x01)
				{
					// Reset the current period counter
					currentPeriod = 0;
					if(next == 100)
					{
						fail = 0;
						success = 1;
						periodCounter--;
					}
				}
				else
				{
```

```
                    tc->Channel_0.CCR = 0x05;
                }

        }
        else
                if(buttonState == BUT_PRESSED)
                buttonState = BUT_IDLE;
        }
}

if(fiq & (1<<TC0_ID))
{
        data_out = tc->Channel_0.SR;
        aic->ICCR = (1<<TC0_ID);
        data_out = pioa->ODSR;

        // Increase the period counter
        currentPeriod++;

        // If the maximum number of periodCounter has been reached
        if(currentPeriod == periodCounter)
        {

                currentPeriod = 0;

                if(next == 100)
                {
                        status = 0;
                        fail++;
```

```
                // If there are 3 fails
                if(fail == 3)
                {
                        fail = 0;
                        success = 1;
                        next = 100;
                        status = 0;
                        periodCounter = 20;
                        currentNext = 0;
                        tc->Channel_0.CCR = 0x02;
                        fail_message = 1;
                }
        }
        else
        {
                // Start counting ascending
                if(next == 0)
                {
                        status = 1;
                }
        }

        // Set the next
        if(status)
                next++;
        else
                next--;
}
```

```
                else
                        // Reset the success flag
                        if(success)
                        {
                                success = 0;
                        }

                        // Reset the clock
                        tc->Channel_0.CCR = 0x05;
                }
        }
}
```