Intermediate Project
Final report

Propeller Clock

Przemysław Bartosik Michał Zasłona

Supervisor: Dr. Witold Paluszyński

Department of Cybernetics and Robotics
Wrocław University of Technology

February 15, 2016

Abstract

The main goal of this project was to build a propeller clock. The authors assumed using Bluetooth communication protocol as a way of communication between a user and the clock. There were also assumed two different power supply sources: from battery (to power the propeller) and from DC power supply (to run a DC motor which spins the propeller). The project has not been finished yet, but most of the work has been done. The clock displays a clock face with properly working minute and hour hands. The calibration of the image is still slightly moved, but it is legible. However, there are still two tasks left to complete the project: Bluetooth communication and finishing GUI application.

1 Introduction

The main goal of this project was to build a propeller clock. The general idea of a propeller clock is to give the illusion of a 'screen' by rotation in high speed blinking LEDs. A motor spins the "propeller" and a small microprocessor keeps track of time and changes the pattern on 16 LEDs with exact timing to simulate a clock dial with its hands. The result of such construction may be very effective and eye-catching. The inspiration for this topic were videos found online of propeller clocks made by different people.

The main assumption of this project was to build a propeller clock that would not only be showing a current time but also date and some short, user-defined sentences. The way of determining the appearance of the clock and its functionality at a certain moment was planned to be a GUI application using Bluetooth communication protocol. The initial plan was also to use STM32 processor on a Discovery VL board. However, that plan was quickly given up because of the weight and size of the whole board in favor of a simple AVR microcontroller ATmega328.

The construction may be divided into two main parts:

1. the propeller – there is located a microcontroller which controls the speed of LEDs, handles Bluetooth connection (using a Bluetooth module) and uses a phototransistor in order to measure the speed of the propeller,

2. the base – it uses DC motor to spin the propeller and a diode IR to give the phototransistor a point of reference (more details in **Section 3**).

2 The construction

The mechanical part turned out to be a crucial part of the project. Especially, as for the construction of the propeller, several problems were hard to overcome. When designing a propeller (which was simply a PCB board), the following criteria were taken into account:

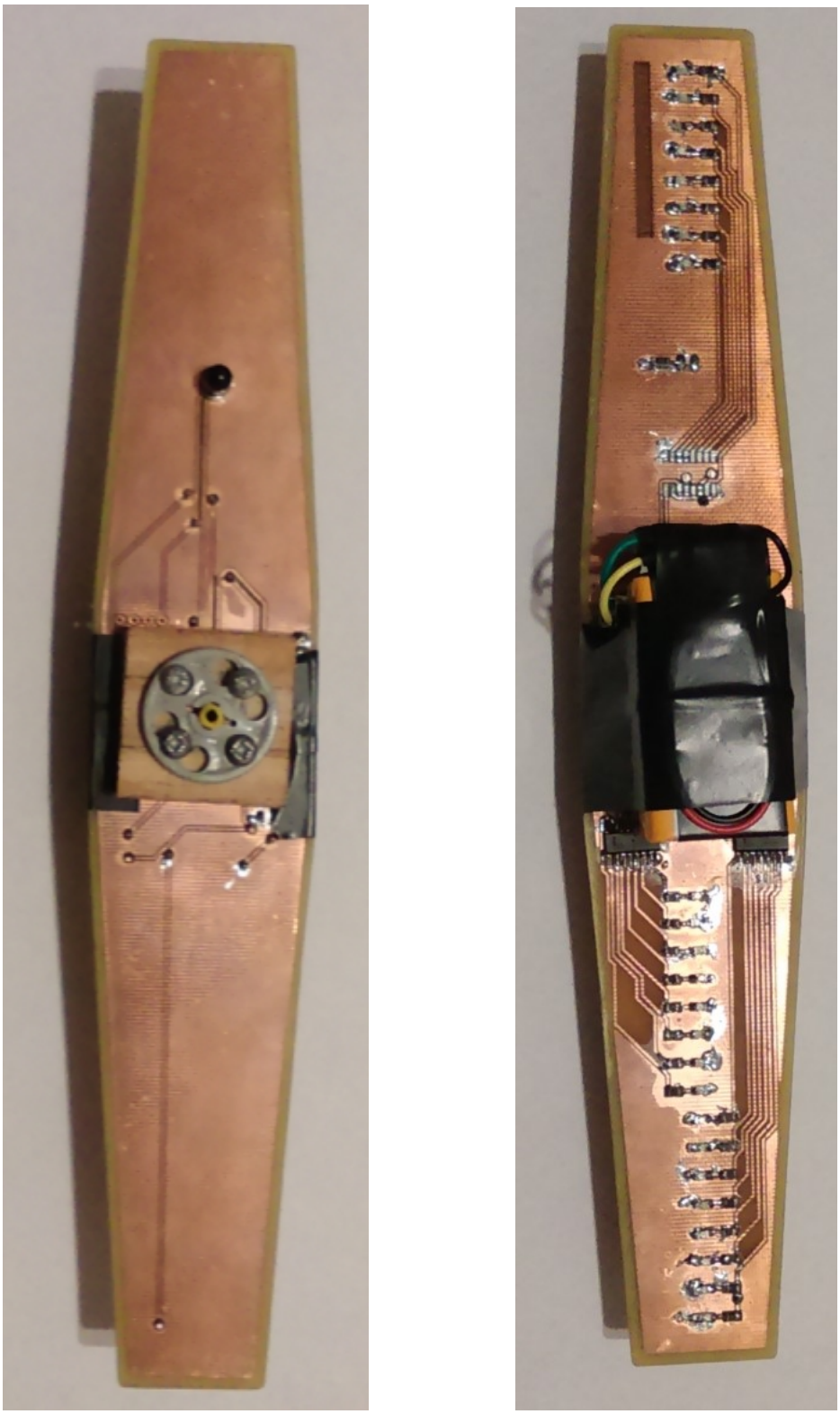
- the size – the smaller a propeller is, the better because bigger constructions might have worse aerodynamics which may create more power consumption,

- the weight – if the propeller is too heavy, it may be harder to balance it while it is moving which may cause major problems (e.g. oscillation),

- LEDs arrangement – it is important that the LEDs are not too far from each other and are in the same row (especially when they are on both sides of the propeller; in that case they should be also symmetrical relative to each other),

- heavy parts arrangement – the propeller may (but should not) consist of some heavy parts; it is important to make sure that they are located in the right place taking into account the center of the mass – in case of this project, the center of the mass was located right in the middle of the propeller.

Another issue was a proper assembling of a propeller with a shaft of a DC motor. The solution turned out to be two wooden blocks with a small hole on one side which the shaft could fit. These blocks have been attached on the other side to the propeller using four 2cm screws. In order to isolate screw nut from the copper fields, there were used plastic washers made of LEGO parts.

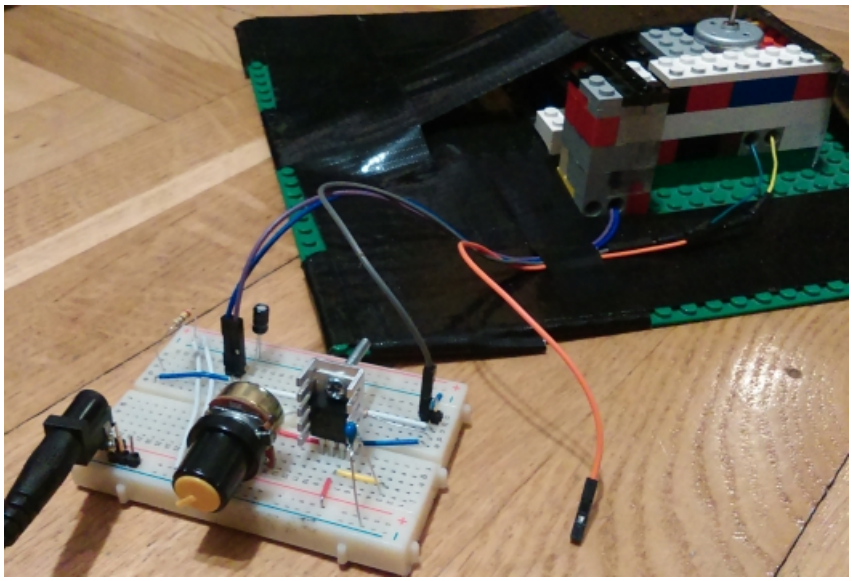


(a) bottom layer, wooden block and small whole for a shaft of DC motor

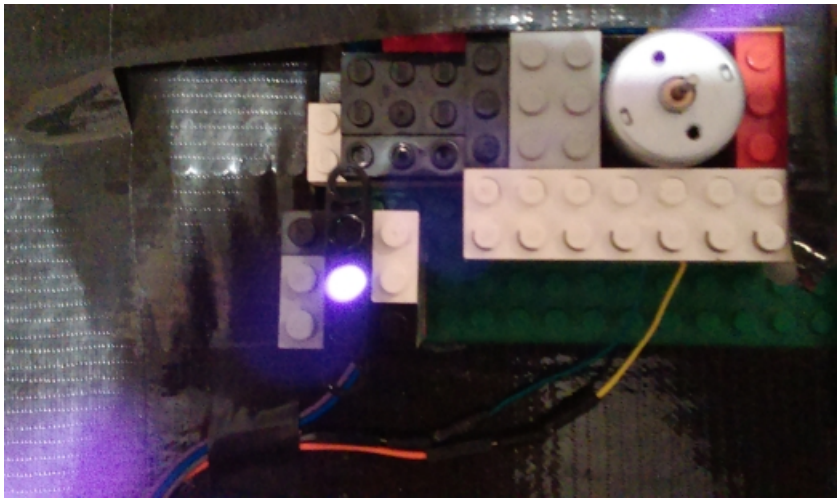
(b) top layer, battery attached on geometric center for greater stability

Figure 1: Construction of propeller

The base for a DC motor is also made of LEGO blocks. The reason why this kind of construction was applied was its simplicity to make and great availability of parts. What is even more important, the DC motor fits in the blocks very tightly which creates a great stability.



(a) voltage regulator for DC motor and LEGO base in background



(b) top view on LEGO base, depicted DC motor and IR diode (lightening)

Figure 2: Construction of base for DC motor

3 Electronics

The most of electronic parts have been located on a propeller. It includes:

- an AVR microcontroller (ATmega328P-AU),
- 2 shift registers 74HC595 in SMD package (one per each 8 LEDs) (initially there were 3 of them, as visible in **Figure 3**),
- 16 LEDs in SMD (with SMD resistors) (initially there were 24 of them, as visible in **Figure 3**),
- a Bluetooth module,
- a phototransistor in THT package,

- a 9V battery.

When choosing these components, two main criteria were taken into account in all cases: its size and weight. In order to get the lightest and the smallest parts, most of electronic parts are in SMD package.

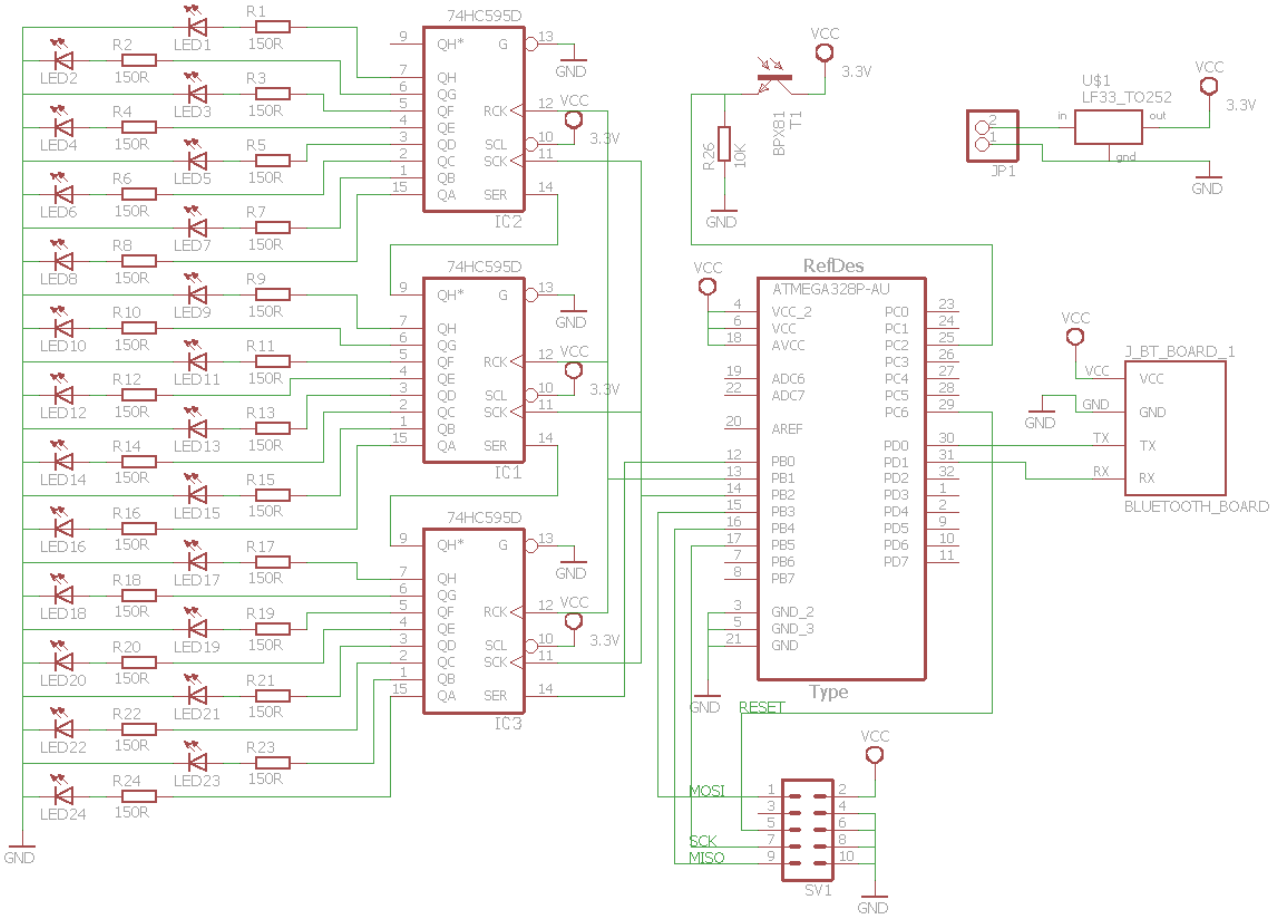
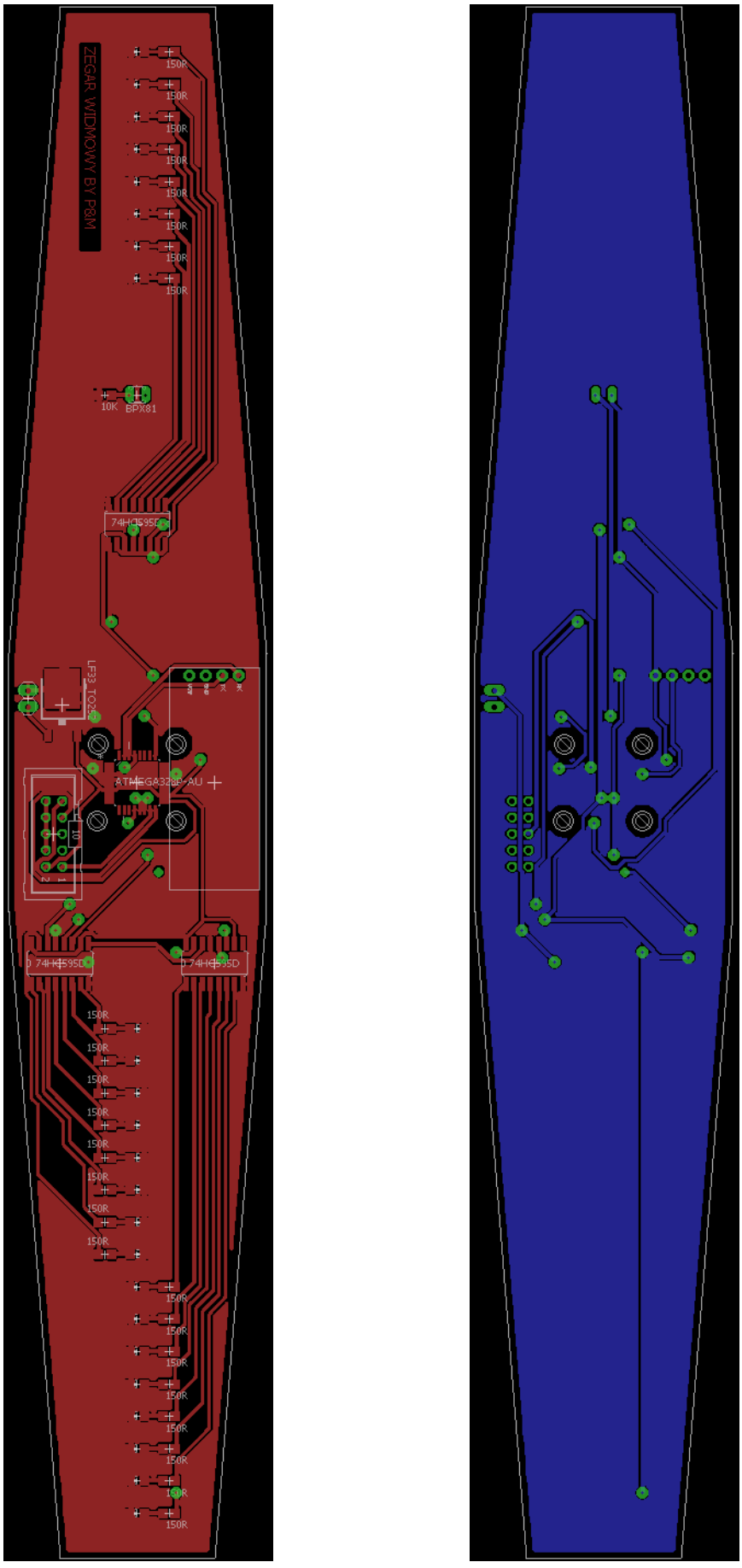


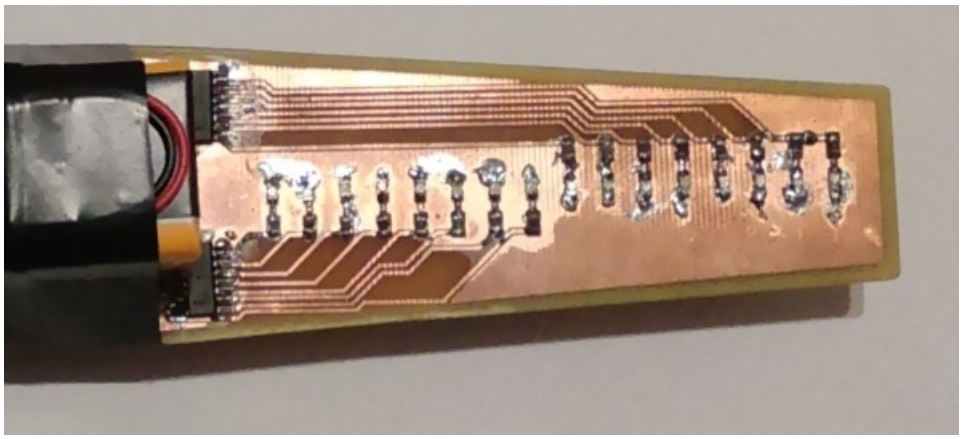
Figure 3: The PCB schematics made in Cadsoft Eagle



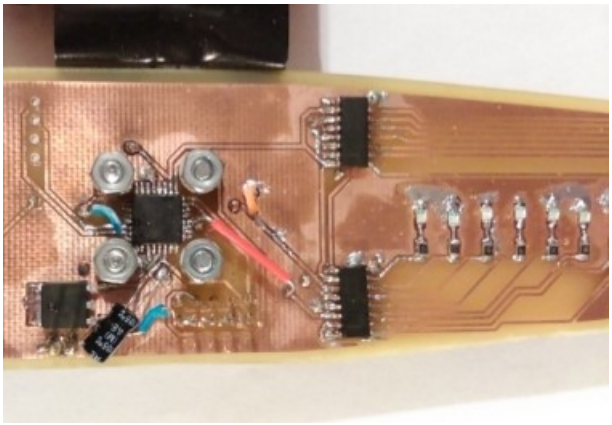
(a) a top layer (b) a bottom layer

Figure 4: The PCB project made in Cadsoft Eagle

As for the electronics, the most problematic issue was the power source for the propeller. In different projects of propeller clocks the approach is slightly different. In some of them there are used, so-called 'brushes' in order to pass current through the shaft of a DC motor to a spinning propeller. However, that kind of solution requires a great precision in construction. If the construction is faulty, it may may cause breaks in power supply (which may be result in constant resetting of a microcontroller) or even sparking between brushes which may be dangerous for the system. In this project the preferred power supply was a 9V battery located in the center of the propeller. The main reason for using it was simplicity in connecting it to the microcontroller and its reliability as a power source. However, that solution has several drawbacks as well. The major problem is the size of the battery an its weight.



(a) view on propeller - battery, 2 shift registers and 16 leds



(b) 'heart' of propeller (ATmega328P) and two shift registers which controls LEDs

Figure 5: Main components on TOP propeller side

Another essential task to accomplish was getting information about the speed of the propeller. The solution in this case was rather simple. The authors decided to use for that purpose an IR diode with a phototransistor. The IR diode is mounted to the base of the propeller (presented on **Figure 2b**) and it is used as a reference point for the phototransistor. The phototransistor, mounted on the bottom side of the propeller (presented on **Figure 1a**, is spinning with the propeller while the IR diode remains static. Whenever they meet, the phototransistor drives high (logical 1) which is registered by the microcontroller. By counting ticks in a specified time range, the microcontroller is able to determine the frequency of the spinning propeller which is used for synchronization of LEDs with the speed of the propeller.

Controlling the speed of the propeller was possible by using a simple, adjustible voltage controller. The construction of it consists of:

- 3.0A, adjustable output, positive voltage regulator LM350,
- linear potentiometer 5KΩ,
- resistor 240Ω,
- capacitor 0.1μF,
- capacitor 1μF,
- a breadboard.

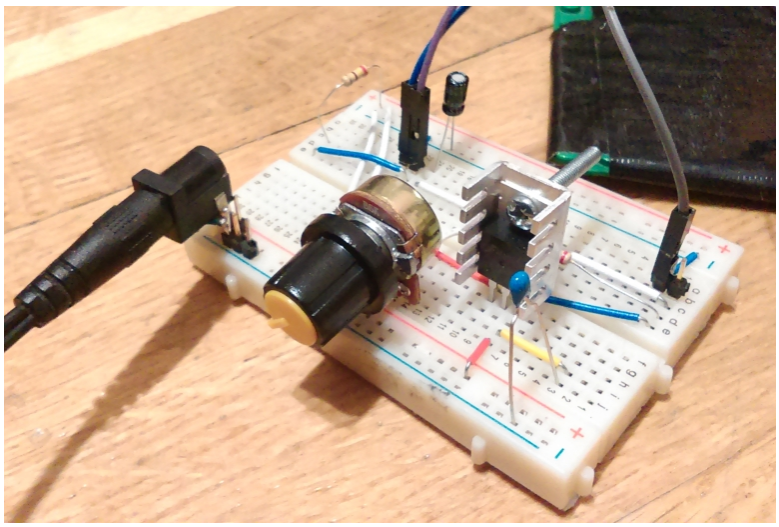


Figure 6: Simple voltage controller, constructed to control speed of DC motor.

4 The program

4.1 The interrupts

The program handles two interrupts. First of them (`ISR(INT0_vect)`) is triggered by a rising edge on pin PD2 and it is responsible for counting rotations of the propeller. Second interrupt is coupled with a timer - it is triggered every 6 seconds. Its purpose is to:

- compute the period T of rotation of the propeller (it is an average of the old period T_{old} and new period T_{new}),
- determine the position of a minute and hour hands of the clock,

4.2 Algorithm of displaying a dial

1. Calculate period (based on interrupt from phototransistor).
2. Calculate delays between each patterns to be displayed for each part of clock dial.
3. Set reset flag which is responsible for starting drawing from exactly the same point, each time (at the moment of interrupt from phototransistor).
4. Turn on and turn off a proper segment of dial.

The whole drawing cycle should last no more than the period of rotation of the propeller. Otherwise, some cycles may be lost which causes blinking of the screen (the image is not smooth).

4.3 The code

The program was written in C programming language. The compiler used for the code was AVR-GCC. The programming environment chosen for the purpose of code development was WinAVR (version: 20100110).

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define DS_PORT    PORTB
#define DS_PIN     0
#define ST_CP_PORT PORTB
#define ST_CP_PIN  1
#define SH_CP_PORT PORTB
#define SH_CP_PIN  2

#define DS_low()  DS_PORT&=~_BV(DS_PIN)
#define DS_high() DS_PORT|=_BV(DS_PIN)
#define ST_CP_low()  ST_CP_PORT&=~_BV(ST_CP_PIN)
#define ST_CP_high() ST_CP_PORT|=_BV(ST_CP_PIN)
#define SH_CP_low()  SH_CP_PORT&=~_BV(SH_CP_PIN)
#define SH_CP_high() SH_CP_PORT|=_BV(SH_CP_PIN)

//Define functions
//================================================
void ioinit(void);
// It initializes GPIO ports
// and turns on the interrupt INT0

void timer_init(void);
// It initializes timer interrupt OCR1A in CTC mode 4

void output_led_state(unsigned long int __led_state);
// It handles displaying LED pattern using two shift registers

void draw_clock_face(void);
// It displays in a proper way a clock dial

void draw_long_line(double small_delay);
// It displays an hour scale of a clock dial

void draw_short_line(double small_delay);
// It displays a minute scale of a clock dial
```

```
void draw_minute_line(double small_delay);
// It displays a minute hand of a clock

void draw_hour_line(double small_delay);
// It displays an hour hand of a clock

//=============================================

//Define global variables

volatile double period = 500000.0;
// period of a single rotation (us); starting value - 500000us

volatile unsigned char rot_num = 0;
// number of rotations counted by a phototransistor;
// the number is reset by timer interrupt (after computing frequency)

double counting_time = 6000000.0;
// the time of counting rotations (us);
// after that time there is an interrupt

volatile double delay = 500000.0;
// delay used in function _delay_us()

volatile double current_period = 0;        // current period

volatile double old_period = 0;            // old period

volatile unsigned char flag = 0;           // flag for the interrupt

volatile unsigned char hour_pos = 2;       // position of hour line 0-11

volatile unsigned char minute_pos = 58;    // position of minute line 0-59

volatile unsigned char timer_count_minute = 0;
// a counter used in timer interrupt (0-10);
// after every 6s (triggering interrupt), the counter is incremented;
// when the number 10 is reached (60s), the minute hand of the clock
// changes its position

char led_num[16] = {7,0,1,2,3,4,5,6,15,14,13,12,11,10,9,8};
// array which stores a proper order of LEDs from 1 to 16

unsigned int five_minute_line = 0; // LED pattern for hour scale on a dial

unsigned int one_minute_line = 0;  // LED pattern for minute scale on a dial

unsigned int hour_line = 0;        // LED pattern for an hour hand of a clock
```

```
int main (void)
{
    timer_init(); // setup timer
    ioinit(); //Setup IO pins and defaults

    five_minute_line = _BV(led_num[0]) | _BV(led_num[1])
                     | _BV(led_num[2]) | _BV(led_num[3]);
    one_minute_line = _BV(led_num[0]);
    hour_line = 0xFF00 | five_minute_line;

    while(1)
    {
        if(flag == 1)
        {
            draw_clock_face();
            flag = 0;
        }
    }
}


void ioinit (void)
{
    DDRB  = 0b00000111; //1 = output, 0 = input
    PORTB = 0b00000000;
    DDRD  = 0b11111011;   // set PD2 to input
    PORTD = 0b00000100;   // set PD2 to high

    EICRA |= (1<<ISC01) | (1<<ISC00);
    // rising edge on INT0 generates interrupt
    EIMSK |= (1<<INT0);
    // turn on interrupt INT0
    sei();
    // turn on interrupts
}

void output_led_state(unsigned long int __led_state)
{
   SH_CP_low();
   ST_CP_low();
   for (unsigned long int i=0;i<16;i++)
   {
      if((_BV(i) & __led_state) == _BV(i))
         DS_high();
      else
         DS_low();


      SH_CP_high();
      SH_CP_low();
```

```
    }
    ST_CP_high();
}


void timer_init(void)
{
    // OCRn = (8000000 / 1024)*6-1 = 46874
    // desired time = 6s
    // 8MHz, 1024 - prescaler
    // OCR1A = 46874 = 0xB71A

    OCR1A = 0xB71A;

    TCCR1B |= (1 << WGM12);
    // Mode 4, CTC on OCR1A

    TIMSK1 |= (1 << OCIE1A);
    //Set interrupt on compare match

    TCCR1B |= (1 << CS12) | (1 << CS10);
    // set prescaler to 1024 and start the timer
}


void draw_clock_face(void)
{
    unsigned char i = 0;
    unsigned char num_of_parts = 60;
    double part_delay = 0.97*delay/num_of_parts;
    double small_delay = 0.03*delay/num_of_parts;

    for(i=0; i<num_of_parts; i++)
    {
        if(i == minute_pos)
            draw_minute_line(small_delay);
        else if(i%5 == 0)
        {
            if(i == hour_pos*5)
                draw_hour_line(small_delay);
            else
                draw_long_line(small_delay);
        }
        else
            draw_short_line(small_delay);

        _delay_us(part_delay);
    }
}
```

```
void draw_long_line(double small_delay)
{
    output_led_state(five_minute_line);
    _delay_us(small_delay);
    output_led_state(0x0000);
}

void draw_short_line(double small_delay)
{
    output_led_state(one_minute_line);
    _delay_us(small_delay);
    output_led_state(0x0000);
}

void draw_minute_line(double small_delay)
{
    output_led_state(0xFFFF);
    _delay_us(small_delay);
    output_led_state(0x0000);
}

void draw_hour_line(double small_delay)
{
    output_led_state(hour_line);
    _delay_us(small_delay);
    output_led_state(0x0000);
}

// This interrupt is triggered by rising edge on PD2
// (when the phototransistor meets IR diode)
// It counts number of rotations per every 6s.
ISR (INT0_vect)
{
    rot_num++; // phototransistor has detected IR diode
    flag = 1;
}

// This interrupt is triggered every 6s. It handles:
// - computing the current period
// - computing a proper delay for the whole cycle
//   (including computation processes of the microcontroller)
// - determining position of minute and hour hands of the clock
ISR (TIMER1_COMPA_vect)
{
    if(rot_num > 10)
    {
        current_period = counting_time / rot_num;
        // compute current period in us
        period = (old_period+current_period)/2;
```

```
        // period is the average of old and current
    }
    else if(rot_num != 0)
    {
        period = counting_time / rot_num; // compute period in us
    }
    else
    {
        period = 500000.0;
    }

    timer_count_minute++;

    if(timer_count_minute == 10){
        if(minute_pos < 59)
        {
            minute_pos++;
        }
        else
        {
            minute_pos = 0;
            hour_pos++;
            if(hour_pos == 11)
                hour_pos = 0;
        }
        timer_count_minute = 0;
    }

    delay = 0.753*period;        // the period for draw line is smaller
    rot_num = 0;            // reset rot_num
    old_period = period;
}
```

5 Tests

In order to decide whether Bluetooth transmission between a computer and a spinning propeller is possible, there has been assembled a special construction of a propeller for tests which consists of:

- a small breadboard,
- a Bluetooth module,
- an AVR microcontroller (ATmega8),
- a 9V battery,
- a voltage regulator (LM7805)

and a DC motor which enables the propeller to spin.

The microcontroller had loaded into memory a program that receives (via Bluetooth) two integers and then it sends back the result of the linear equation with these two integers as the coefficients (respectively: a,b). The procedure of the test is presented below.

1. Power the microcontroller (give it power supply from the battery).
2. Power the DC motor which makes the propeller (a breadboard) spin.
3. Run a serial terminal program (such as RealTerm) and connect to a Bluetooth module.
4. Once there is connection between two of them, send two numbers.
5. The result of the linear equation should be visible now in the serial terminal program.

The tests turned out to be successful which confirmed that it is in fact possible to connect to a Bluetooth module attached to a spinning propeller.

Also two tests additionally were made: test of controlling LEDs using shift registers and test of work of phototransistor with IR diode.

6 Summary

The project has not been completely finished yet. Most of work has been done, separate parts of the project were realized. The most important task was to connect divided parts of the project which were done separately. Main part - program that controls the LEDs depending on the speed of the propeller is done. Features and properties of designed device are not exactly the same as it was assumed, but obtained results (presented on **Figure 7)** are satisfying.

The clock displays a clock face with properly working minute and hour hands. The calibration of the image is still slightly moved, but it is legible.
However, two task left to complete the project:

- Bluetooth communication,
- finishing GUI application so that it can be used for easy control of the clock.

7 References

- Kolorowy obrotowy wyświetlacz widmowy (PL)
- Introduction to 74HC595 shift registers
- The Propeller Clock (YouTube video)
- Propellerclock - Ring Of Fire (YouTube video)

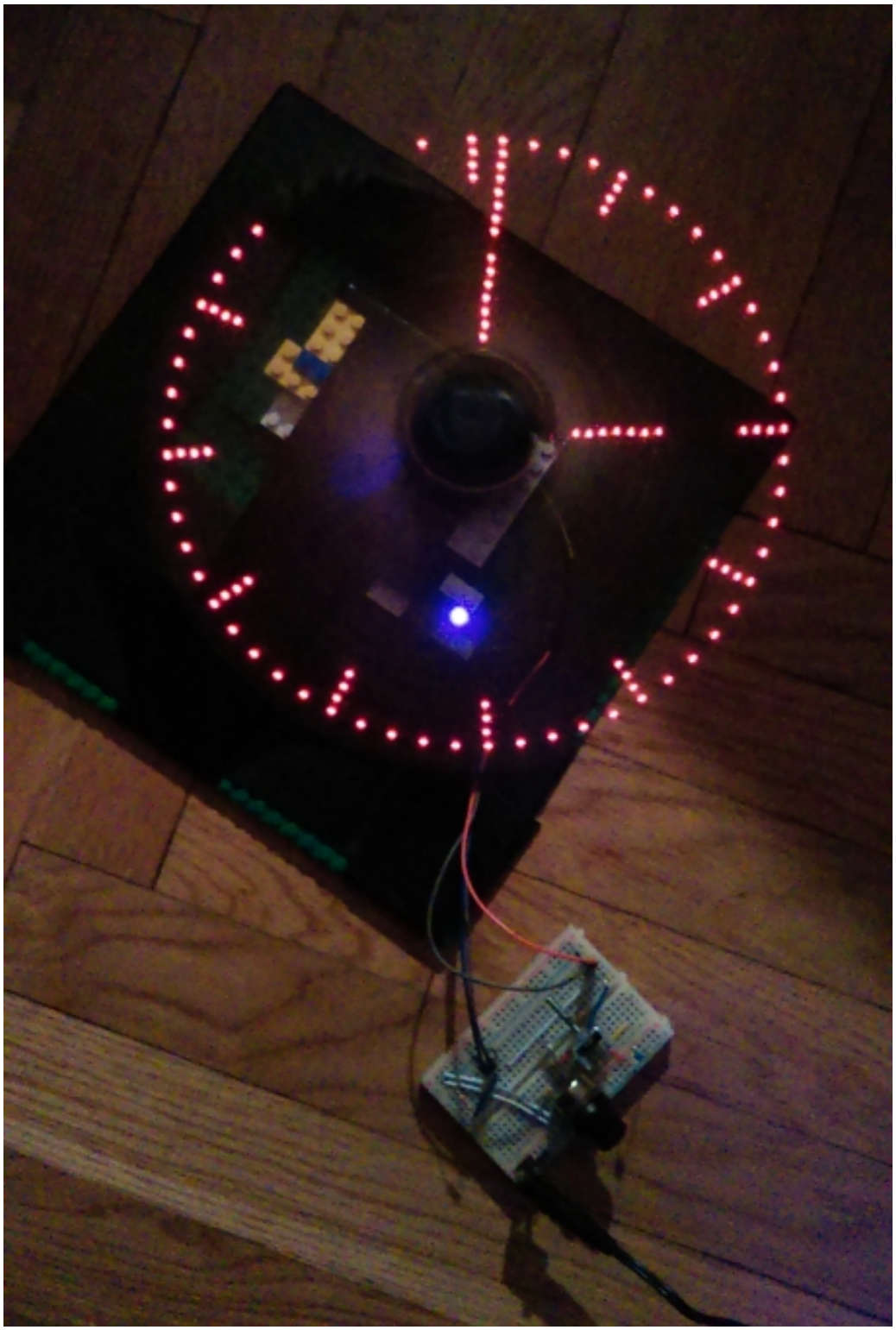


Figure 7: Final effect