Eight-Segment Display controlled with shift register and Arduino Uno

1 Operating instructions and description of the device

The eight-segment horizontal display is a device designed to represent read data, displaying conventional characters, digits and binary numbers.

The present display was made using a 74HC595 shift register. Each individual segment of the display consists of a rectangle composed of 3mm blue LEDs with a size of 6 x 3. The total number of diodes used is 144.

The power supply for the LEDs is provided by a self-built power supply on an LM337 chip. Its theoretical maximum power is 20W, voltage regulated from 0 to 12V.

The Arduino Uno microcontroller is used to send instructions according to data from other peripherals (100k ohm linear potentiometer, numeric keypad, distance sensor) to the 74HC595 shift register.

To the outputs of the shift register are connected bases of BD139 transistors, which in this circuit act as keys for the 5V supply coming from a regulated power supply.

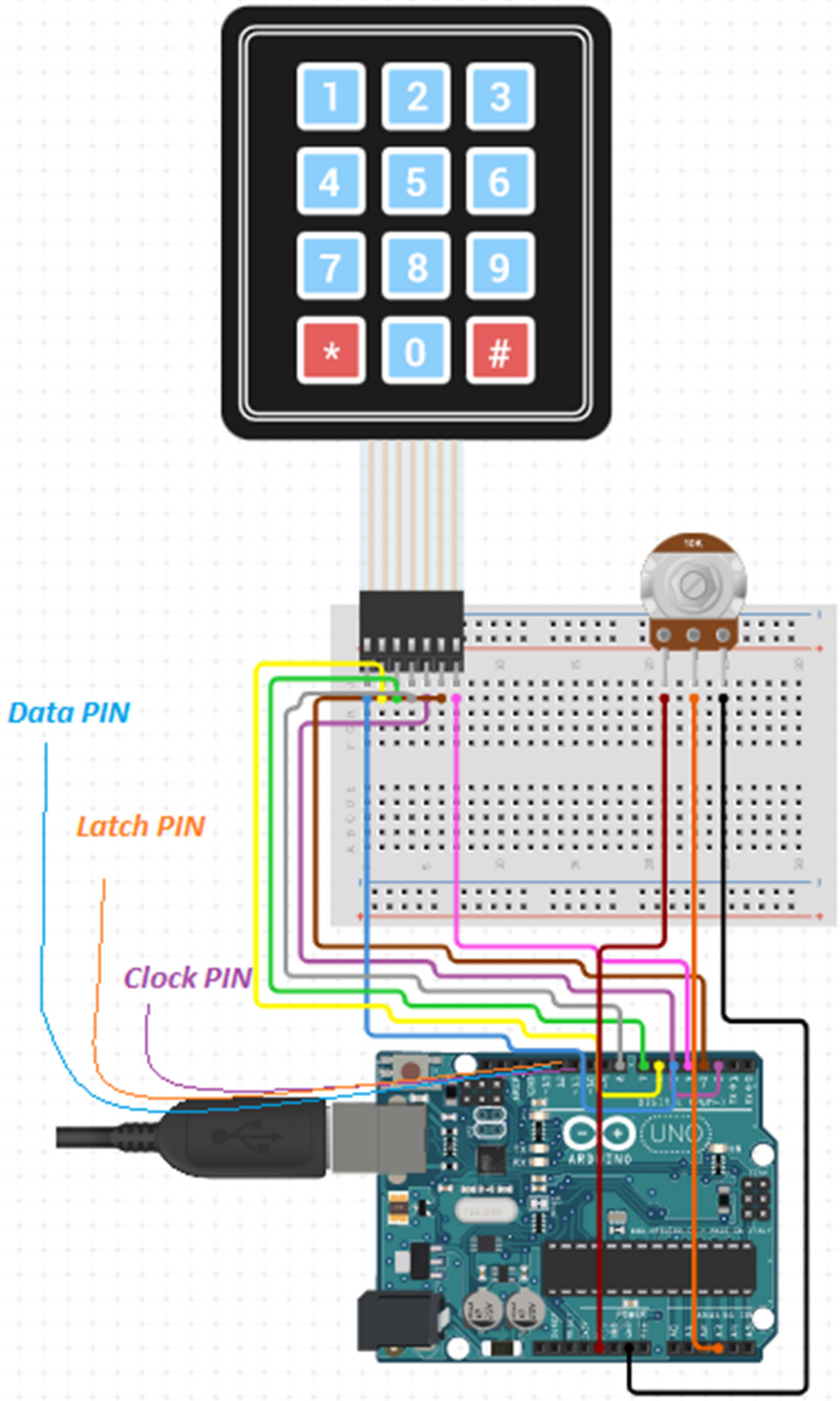
To start the device, you need to connect the microcontroller power supply and the LED power supply (minus the power supply to the emitters of the control transistors, while plus to the anodes of the LEDs).

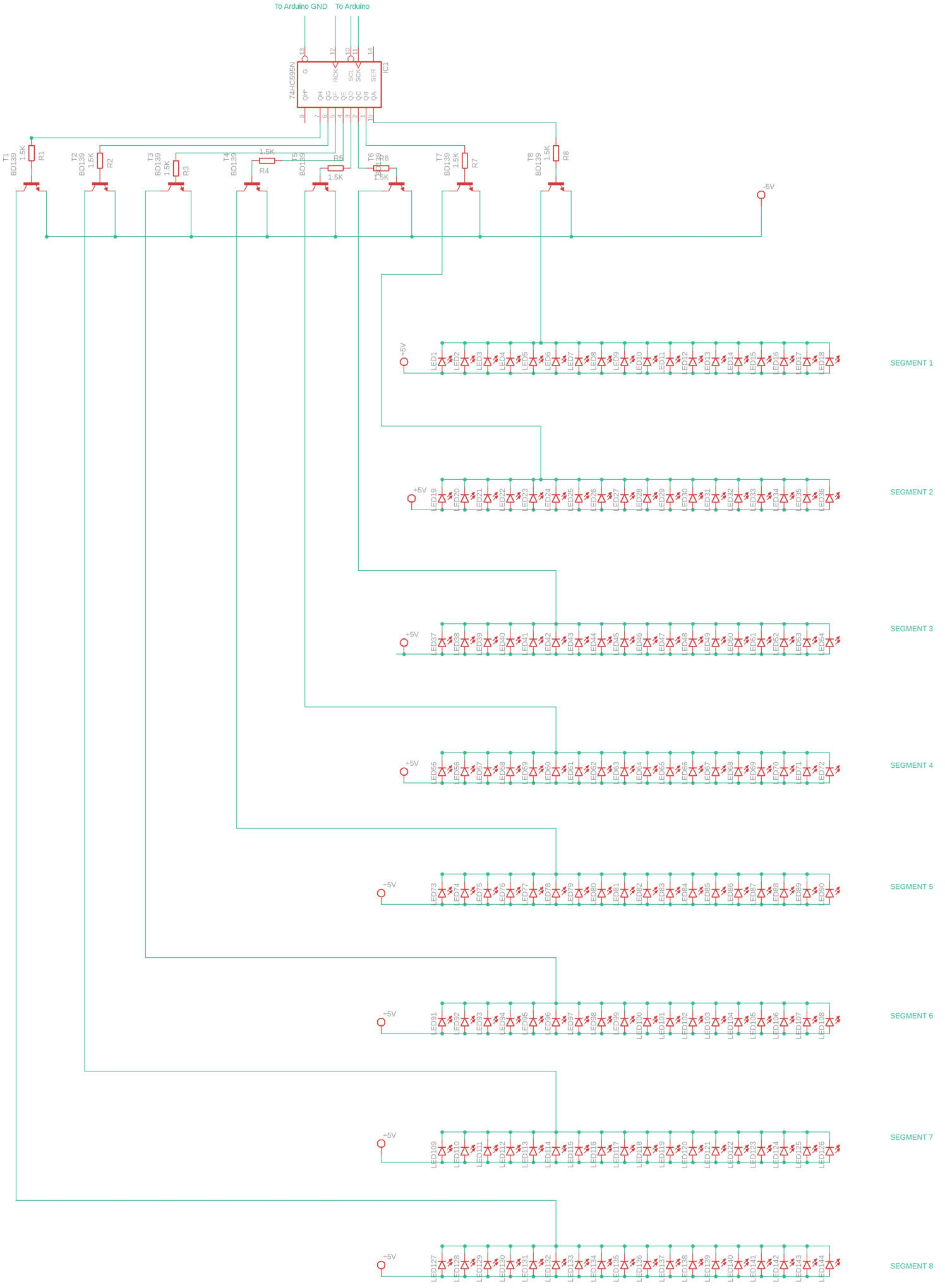
In order to demonstrate the operation of the display, several codes have been created, which should be implemented through the Arduino IDE environment to the programmer of the arduino UNO board via the USB port.

Do not exceed 5V on the regulated power supply. Exceeding this voltage risks burning out the diodes completely.

The device is not suitable for operation at room temperature for more than 15 minutes due to too weak thermal convection of transistors (no additional cooling in the circuit).

SCHEMATIC




To Arduino GND
To Arduino
74HC595N
IC1
T1 BD139 1.5K R1
T2 BD139 1.5K R2
T3 BD139 1.5K R3
T4 BD139 R4 1.5K
T5 BD139 R5 1.5K
T6 BD139 R6 1.5K
T7 BD139 1.5K R7
T8 BD139 1.5K R8
-5V
+5V
SEGMENT 1
SEGMENT 2
SEGMENT 3
SEGMENT 4
SEGMENT 5
SEGMENT 6
SEGMENT 7
SEGMENT 8

CODE 1 - TURNING ON SEGMENTS AUTOMATICALLY

```
int clockPin = 11;
int latchPin = 12;
int dataPin = 13;
byte leds = 0;
byte screen;
byte sum;
void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  Serial.begin(57600);
}
void loop()
{
  updateShiftRegister();
  delay(200);
  for (int i = 0; i < 8; i++)
  {
    bitSet(leds, i);
    updateShiftRegister();
    delay(200);
  }
  for (int i = 8; i>=0;i--)
  {
    sum=pow(2,i);
    screen= pow(2,i-1);
    Serial.print("screen");
    Serial.print(screen);
```

```
leds = sum;
 bitSet(leds, i-1);
updateShiftRegister();
}
}
void updateShiftRegister()
{
  delay(1000);
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}
```

CODE 2 - TURNING ON SEGMENTS MANUALLY

```
#define trigPin 3
#define echoPin 2
int clockPin = 11;
int latchPin = 12;
int dataPin = 13;
byte leds = 0;
byte screen;
byte sum;
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;
byte rowPins[ROWS] = {2, 3, 4, 5};
byte colPins[COLS] = {6, 7, 8, 9};
```

```
char keys[ROWS][COLS] =
 {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};

Keypad klawiatura = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  char klawisz = klawiatura.getKey();

  if (klawisz=='1'){
    Serial.println(klawisz);
    updateShiftRegister();
    leds = 1;

  }
  else if (klawisz=='2')
  {
  Serial.println(klawisz);
  updateShiftRegister();
  leds=2;
```

```
  }
  else if (klawisz=='3')
  {
  Serial.println(klawisz);
  updateShiftRegister();
  leds=4;

  }
  else if (klawisz=='4')
  {
  Serial.println(klawisz);
  updateShiftRegister();
  leds=8;

  }
else if (klawisz=='5')
  {
  Serial.println(klawisz);
  updateShiftRegister();
  leds= 16;

  }
  else if (klawisz=='6')
  {
  Serial.println(klawisz);
  updateShiftRegister();
  leds =32;

  }
  else if (klawisz=='7')
```

```
  {
  Serial.println(klawisz);
  updateShiftRegister();
  leds= 64;
  }
else if (klawisz=='8')
  {
  Serial.println(klawisz);
  updateShiftRegister();
  leds =128;
  }
  else if (klawisz=='9')
  {
  Serial.println(klawisz);
  updateShiftRegister();
  leds = 255;
  }
  else if (klawisz=='0')
  {
  Serial.println(klawisz);
  updateShiftRegister();
  leds = 0;
}
void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
     digitalWrite(latchPin, HIGH);
}
```

CODE 3 - SWITCHING OF DIFFERENT DISPLAY MODES AND ADJUSTMENT OF TIME SETTING BY POTENTIOMETER

```
#define trigPin 3
#define echoPin 2
int clockPin = 11;
int latchPin = 12;
int dataPin = 13;
byte leds = 0;
byte screen=0;
byte sum=0;
#include <Keypad.h> //biblioteka od klawiatury
int ADC_time= 0;
int i=0;
int Time=0;
long randnumber=0;
const byte ROWS = 4; // ile wierszy
const byte COLS = 4; //ile kolumn

byte rowPins[ROWS] = {2, 3, 4, 5};
byte colPins[COLS] = {6, 7, 8, 9};

char keys[ROWS][COLS] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}




};
```

```
Keypad klawiatura = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){

  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  char klawisz = klawiatura.getKey();

  if (klawisz=='1'){
    Serial.println(klawisz);
    while (i<=256)
    {
  ADC_time = analogRead(A5);
  Serial.println(ADC_time);
      i++;
    updateShiftRegister();
    leds = i;
    Time=((ADC_time+1)/(100));
    Serial.println(Time);
    delay(Time*100);
    }
    i=0;
  }
  else if (klawisz=='2')
  {
    i=256;
```

```
 while (i>=0)
   {

 ADC_time = analogRead(A5);
    i--;
   updateShiftRegister();
   leds = i;
   Time=((ADC_time+1)/(100));
   Serial.println(Time);
   delay(Time*100);
   }
 }
 else if (klawisz=='3')
 {
  i=256;
 while (i>=0)
   {
 ADC_time = analogRead(A5);
    i--;
   Time=((ADC_time+1)/(100));
   Serial.println(Time);
   delay(Time*100);
   if (i%2==0)
   {
     updateShiftRegister();
    leds = 240;
   }
   else
 {
     updateShiftRegister();
leds = 15;
```

```
   }
   }



 }
 else if (klawisz=='4')
 {
 i=256;
while (i>=0)
  {



 ADC_time = analogRead(A5);
 Serial.println(ADC_time);
    i--;
  Time=((ADC_time+1)/(100));
  Serial.println(Time);
  delay(Time*100);
  if (i%2==0)
  {
     updateShiftRegister();
    leds = 255;
  }
  else
  {
     updateShiftRegister();
leds = 0;
  }
  }
 }
else if (klawisz=='5')
```

```
  {
    Time=0;
    randnumber=0;
    i==0;
    while(i<=100)
    {
  ADC_time = analogRead(A5);
  Serial.println(Time);
  Time=((ADC_time)/(4.018));
  delay(100);
  leds = Time+1;
     updateShiftRegister();
  i++;
    }
  }
  else if (klawisz=='6')
  {
    i=0;
    randnumber=0;
      while(i<=12)
    {
 if (i%2==0)
    {
       updateShiftRegister();
      leds = 255;
    }
    else
    {
      leds = 0;
    }
    i++;
```

```
   delay(300);
   {
     updateShiftRegister();
leds = 0;
   }

   }
   for(i=0;i<1;i++)
   {
 randnumber=random(255);
   }
   Serial.println(randnumber);

 leds =randnumber;
   updateShiftRegister();
    delay(300);
    leds=0;
    updateShiftRegister();
    delay(300);
     leds =randnumber;
    updateShiftRegister();
    delay(300);
   leds=0;
   updateShiftRegister();
   delay(300);
     leds =randnumber;
   updateShiftRegister();
   delay(300);
   leds=0;
      updateShiftRegister();
   delay(300);
```

```
      leds =randnumber;
      updateShiftRegister();


  }
  else if (klawisz=='7')
  {
  i=0;
  randnumber=0;
  randnumber=random(6);
    while(i<=12)
    {
if (i%2==0)
    {
      updateShiftRegister();
     leds = 63;
    }
    else
    {
     leds = 0;
    }
    i++;
    delay(300);
    {
      updateShiftRegister();
leds = 0;
    }


    }


    if(randnumber==0)
```

```
{
leds = 1;
 updateShiftRegister();
 delay(300);
 leds = 0;
  updateShiftRegister();
  delay (300);
  leds = 1;
   updateShiftRegister();
   delay (300);
   leds =0;
    updateShiftRegister();
    delay(300);
     leds = 1;
     updateShiftRegister();
}
else if(randnumber==1)
{
leds = 2;
   updateShiftRegister();
 delay(300);
 leds = 0;
  updateShiftRegister();
  delay (300);
  leds = 2;
   updateShiftRegister();
   delay (300);
   leds =0;
    updateShiftRegister();
    delay(300);
     leds = 2;
```

```
    updateShiftRegister();


}
else if(randnumber==2)
{
leds = 8;
 updateShiftRegister();
 delay(300);
 leds = 0;
  updateShiftRegister();
  delay (300);
  leds = 8;
   updateShiftRegister();
   delay (300);
   leds =0;
    updateShiftRegister();
    delay(300);
     leds = 8;
     updateShiftRegister();
}
else if(randnumber==3)
{
leds= 16;
 updateShiftRegister();
 delay(300);
 leds = 0;
  updateShiftRegister();
  delay (300);
  leds = 16;
   updateShiftRegister();
   delay (300);
```

```
   leds =0;
    updateShiftRegister();
    delay(300);
     leds = 16;
     updateShiftRegister();
}
else if(randnumber==4)
{
leds=32;
updateShiftRegister();
delay(300);
 leds = 0;
  updateShiftRegister();
  delay (300);
  leds = 32;
   updateShiftRegister();
   delay (300);
   leds =0;
    updateShiftRegister();
    delay(300);
     leds = 32;
     updateShiftRegister();
}
else if(randnumber==5)
{
leds = 4;
 updateShiftRegister();
 delay(300);
 leds = 0;
  updateShiftRegister();
  delay (300);
```

```
    leds = 4;
     updateShiftRegister();
     delay (300);
     leds =0;
      updateShiftRegister();
      delay(300);
       leds = 4;
       updateShiftRegister();
  }
 randnumber=0;
 }
else if (klawisz=='8')
 {
 Serial.println(klawisz);
 updateShiftRegister();
 leds =128;

 }
 else if (klawisz=='9')
 {
 Serial.println(klawisz);
 updateShiftRegister();
 leds = 255;

 }
 else if (klawisz=='0')
 {
 Serial.println(klawisz);
 updateShiftRegister();
 leds = 0;
 }
```

```
}
void updateShiftRegister()
{
  digitalWrite(latchPin, LOW);

  shiftOut(dataPin, clockPin, LSBFIRST, leds);

  digitalWrite(latchPin, HIGH);
}
```

CODE 4 - 8 SEGMENT DISPLAY USED AS DISTANCE INDICATOR

```
#define trigPin 3
#define echoPin 2
int clockPin = 11;
int latchPin = 12;
int dataPin = 13;
byte leds = 0;
byte screen;
byte sum;
void setup() {
pinMode(latchPin, OUTPUT);
 pinMode(dataPin, OUTPUT);
 pinMode(clockPin, OUTPUT);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 Serial.begin(9600);
}
```

```
void loop() {
  Serial.print(Distance());
  Serial.println(" cm");

  delay(500);
if (Distance() >=1195)
{

}
else if (Distance() <=10)
{
updateShiftRegister();
leds = 1;
}

else if (Distance()  >=11&& Distance()<=20)

{
updateShiftRegister();
leds = 3;
}
else if (Distance()>=21 && Distance() <=25)

{
updateShiftRegister();
leds = 7;
}
else if (Distance() >=26&& Distance() <=30)
{
updateShiftRegister();
leds = 15;
```

```
}
else if (Distance()>=31 && Distance() <=35)
{
updateShiftRegister();
leds = 31;
}
else if (Distance() >=36 && Distance()<=40)
{
updateShiftRegister();
leds = 63;
}
else if (Distance() >=41 && Distance() <=50)
{
updateShiftRegister();
leds = 127;
}
else if (Distance() >=51)
{
updateShiftRegister();
leds = 255;
}
else
{
updateShiftRegister();
leds = 0;
}
}
void updateShiftRegister()
{
  digitalWrite(latchPin, LOW);
```

```
  shiftOut(dataPin, clockPin, LSBFIRST, leds);

  digitalWrite(latchPin, HIGH);
}

int Distance()
{
 long measured_time, distance;

 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);

 measured_time = pulseIn(echoPin, HIGH);
 distance = measured_time / 58;

 return distance;
}
```