School Physics Experiments with Arduino DUE

Faraday's Law of Induction tells us, that the EMF induced in a coil is directly proportional to the time rate of change of the magnetic flux through the coil.

$$EMF = U_{ind} = -n \cdot \left(\frac{dB}{dt} \cdot A + B \cdot \frac{dA}{dt} \right) = -n \cdot \Phi'(t), \quad \text{where } \Phi(t) \text{ is the magnetic flux through the coil.}$$

The minus sign is Lenz's law.

We perform some experiments with Arduino DUE to determine

① $U_{ind} = U_{ind}(n); \; \frac{dB}{dt} = const; \; A = const.$

② $U_{ind} = U_{ind}\left(\frac{dB}{dt}\right); \; n = const; \; A = const.$

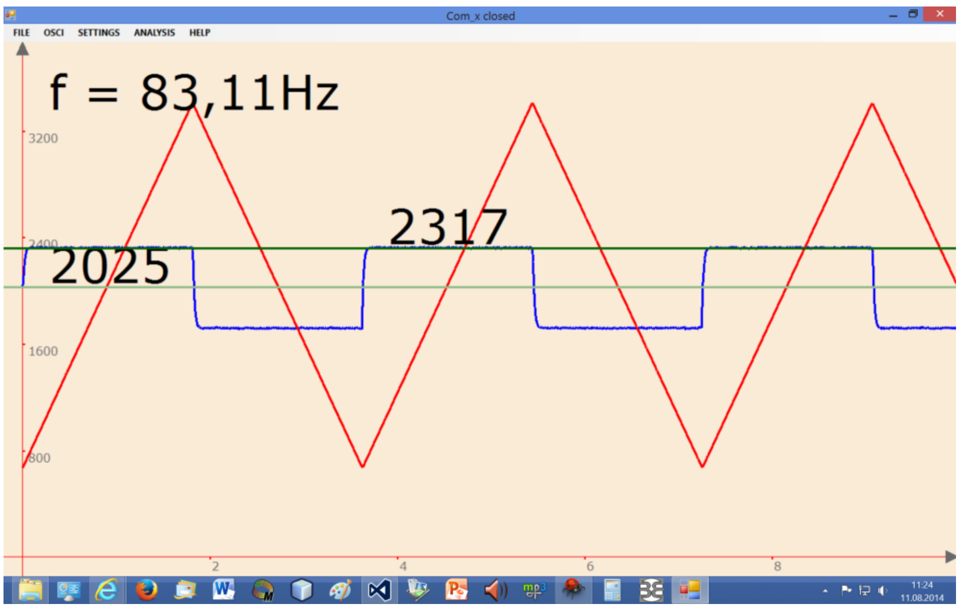
③ $U_{ind} = U_{ind}(A); \; n = const; \; \frac{dB}{dt} = const.$

④ $U_{ind} = U_{ind}\left(\frac{dA}{dt}\right); \; n = const; \; B = const.$

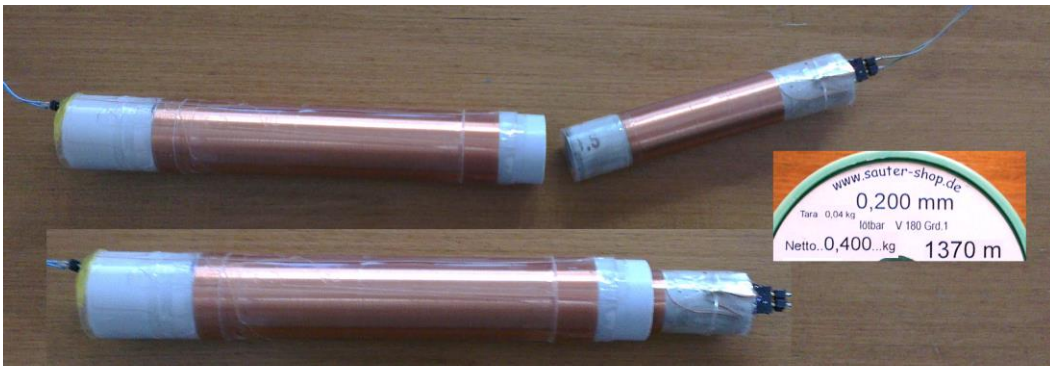
We use for free software Atmel Studio 6.2 with the Visual Micro Tool for Arduino 1.5.x and Visual Studio C# Express for Windows Desktop.

Arduino DUE is (triangle, sin, ...) wave generator and two channel oscilloscope all in one.

Our first ambitious aim shows this picture:



To verify that $U_{ind} = U_{ind}\left(\frac{dB}{dt}\right) = const \cdot \frac{dB}{dt}$; $n = const$; $A = const$; *we need a triangle wave generator and two coils of enamelled copper wire: carrier (plastic pipes) with n=400 windings,* $\varnothing = 20mm$, *n=200 windings,* $\varnothing = 16mm$; *(named: coil400, coil200)*



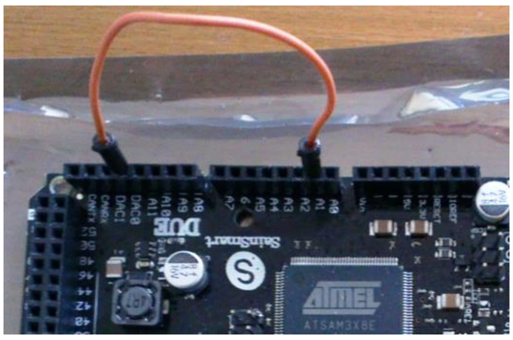
A program for triangle waves for Arduino DUE looks like:

```
int n, dn = 10;

void setup()
{
   analogWriteResolution(12); // 12-bit
}
void loop()
{
   for(n=0; n<4096; n+=dn)
   {
     analogWrite(DAC0, n); //or DAC1
   }
   for(n=4095; n>=0; n-=dn)
   {
     analogWrite(DAC0, n);
   }
}
```

You need an external oscilloscope connected to Pin_DAC0 and Pin_GND to see the triangle wave.

Without such an oscilloscope we have to write a little bit more code and use DUE`s Pin A1.

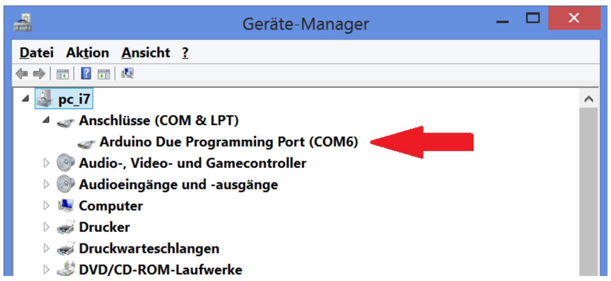
```
int n, dn = 1, ds = 15;
unsigned int i, m;
uint16_t values[24992];
boolean brun = false;
char do_x;
unsigned long t1 = 123456 , t0 = 0;

void setup()
{
   Serial.begin(115200);
   ADC->ADC_MR |= 0x80; //ADC: free running mode
   ADC->ADC_CR=2;
   analogWriteResolution(12); // 12-bit!
}

void loop()
{
```



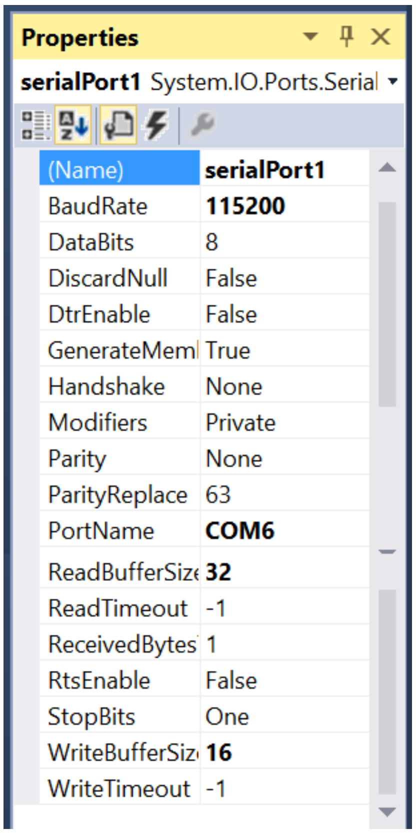
```
   while(Serial.available() > 0) Serial.read();//clear InBuffer
   while(!(Serial.available() > 0)); // wait for SendData
   do_x = Serial.read();
   switch(do_x)
   {
      case 'A':
      ADC->ADC_CHER=0x40;
      delay(100);
      for(n=0; n<500; n++)//for calibration
      {
         analogWrite(DAC0, 0);
         while((ADC->ADC_ISR & 0x40)!=0x40); // wait for conversion
         m=ADC->ADC_CDR[6]; // read data at pin A1
      }//not our measurement
      i=0;
      //begin of measurement
      t0 = micros();
      for(n=0; n<4096; n+=dn)
      {
         analogWrite(DAC0, n);
         while((ADC->ADC_ISR & 0x40)!=0x40); // wait for conversion
         values[i++]=ADC->ADC_CDR[6]; // read data at pin A1
      }
      for(n=4095; n>=0; n-=dn)
      {
         analogWrite(DAC0, n);
         while((ADC->ADC_ISR & 0x40)!=0x40); // wait for conversion
         values[i++]=ADC->ADC_CDR[6]; // read data at pin A1
      }
      t1 = micros();
      //end of measurement
      brun = true;
      while(Serial.available() > 0) Serial.read();//clear InBuffer
      n=0;
      while(brun && n<i)
      {
         Serial.println(values[n]); n+=ds;
         if(n>=i) n=0;
         if(Serial.available() > 0)   brun = false;
      }
      while(Serial.available() > 0) Serial.read();//clear InBuffer "stop_DUE" from C#
      delay(200);
      Serial.println("begin");
      Serial.println(t1 - t0);
      Serial.println("end");
      break;
   };
}
```

A C# oscilloscope program needs the serial port number of a connected Arduino DUE.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```



```
namespace TriWave
{
    public partial class Form1 : Form
    {
        private double xmin, xmax, ymin, ymax, yA = 0, xA = 0;
        private double xStreckfaktor, yStreckfaktor;
        private int npen = 3, df = 10, dn = 1;
        Boolean draw_A1 = false;//channel DUE_A1 is SAM3X8E_channel_A6
        String sDUE = "z";
        double hier_0 = 0;

        public Form1()
        {
            InitializeComponent();
            this.Text = "One or two Channel Oscilloscope";
            xmax = 10;
            xmin = -.2;
            ymax = 4000;//ymax = 1280;
            ymin = -150;
            hier_0 = 1775;
            xA = 0;
            this.DoubleBuffered = false; //faster drawing
            this.WindowState = FormWindowState.Maximized;
        }

        public int xX(double x)
        {
            return (int)Math.Round(xStreckfaktor * (x - xmin));
        }

        public int yY(double y)
        {
            return (int)Math.Round(yStreckfaktor * (ymax - y));
        }

        public double Xx(int x)
        {
            return x / xStreckfaktor + xmin;
        }

        public double Yy(int y)
        {
            return ymax - y / yStreckfaktor;
        }

        public void wait_ms(int dt)
        {
            int t0 = System.Environment.TickCount;
            while (true)
            {
                if (System.Environment.TickCount - t0 > dt) break; //dt milliseconds
            }
        }

        public void Channel_A1(Graphics g, Color c)
        {
            if (serialPort1.IsOpen && draw_A1)
            {
                double x;
                int i, xvon, yvon, xbis, ybis, n_max, mw, ntime = 0;
                Pen pen = new Pen(c, npen);
                Font fn = new Font("Verdana", 48);
                Brush br = new SolidBrush(Color.Black);
                String txt;
                Boolean b_result;
```

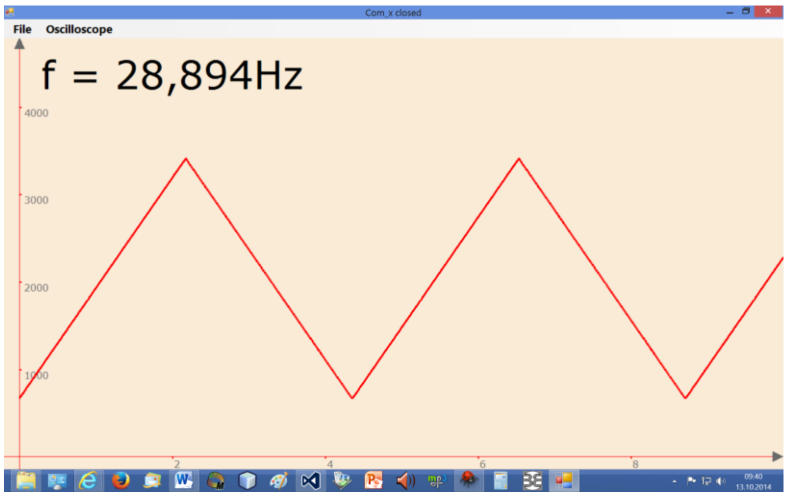


(Name)	serialPort1
BaudRate	115200
DataBits	8
DiscardNull	False
DtrEnable	False
GenerateMeml	True
Handshake	None
Modifiers	Private
Parity	None
ParityReplace	63
PortName	COM6
ReadBufferSiz	32
ReadTimeout	-1
ReceivedBytes	1
RtsEnable	False
StopBits	One
WriteBufferSiz	16
WriteTimeout	-1

```
                x = 0;
                xvon = xX(x); n_max = this.ClientRectangle.Width;
                serialPort1.ReadExisting();
                serialPort1.Write(sDUE);
                wait_ms(200);
                b_result = int.TryParse(serialPort1.ReadLine(), out mw);
                if (b_result)
                    yvon = yY(mw);
                else
                    yvon = yY(0);
                xbis = xvon + dn;
                while (xvon < n_max)
                {
                    b_result = int.TryParse(serialPort1.ReadLine(), out mw);
                    if (b_result)
                        ybis = yY(mw);
                    else
                        ybis = yY(0);
                    g.DrawLine(pen, xvon, yvon, xbis, ybis);
                    xvon = xbis; yvon = ybis;
                    xbis = xvon + dn;
                }
                //--------------------Time-------------
                serialPort1.Write("z");
                serialPort1.ReadExisting();
                wait_ms(100);
                serialPort1.ReadExisting();
                wait_ms(100);
                txt = "";
                for (i = 0; i < 3; i++)
                    txt += serialPort1.ReadLine();
                //g.DrawString(txt, fn, br, 50, 150);
                if (txt.Contains("begin") && txt.Contains("end"))
                {
                    String snum = txt.Remove(0, 5);
                    snum = snum.Remove(snum.IndexOf("end"));
                    b_result = int.TryParse(snum, out ntime);
                }
                if (b_result) g.DrawString("f = " + (1000000.0 / ntime).ToString("0.000") +
"Hz", fn, br, 50, 50);
                //-----------------------------------
                this.BeginInvoke(new EventHandler(closecom));
            }
            draw_A1 = false;
        }

        public void Koordinatensystem(Graphics g, double dx, double dy, double hier_xAchse,
double hier_yAchse)
        {
            Font fn = new Font("Verdana", 12);
            Brush br = new SolidBrush(Color.DimGray);
            Pen pen = new Pen(Color.Red, 1/*Strichdicke*/);

            xStreckfaktor = this.ClientRectangle.Width / (xmax - xmin);
            yStreckfaktor = this.ClientRectangle.Height / (ymax - ymin);
            Point[] pxA =
            {
                new Point(xX(xmax),yY(hier_xAchse)),
                new Point(xX(xmax) - 18,yY(hier_xAchse) - 9),
                new Point(xX(xmax) - 18,yY(hier_xAchse) + 9),
                new Point(xX(xmax),yY(hier_xAchse))
            };
            int h = menuStrip1.Height;
            Point[] pyA =
            {
                new Point(xX(hier_yAchse) + 9, yY(ymax) + 18 + h),
                new Point(xX(hier_yAchse), yY(ymax)-2 + h),
                new Point(xX(hier_yAchse) - 9, yY(ymax) + 18 + h),
                new Point(xX(hier_yAchse) + 9, yY(ymax) + 18 + h)
            };

            g.DrawLine(pen, xX(xmin), yY(hier_xAchse), xX(xmax), yY(hier_xAchse));//x-Achse
            g.FillPolygon(br, pxA);//Pfeil der x-Achse

            g.DrawLine(pen, xX(hier_yAchse), yY(ymin), xX(hier_yAchse), yY(ymax));//y-Achse
            g.FillPolygon(br, pyA);//Pfeil der y-Achse


            double x = dx;
            while (x < xmax)
            {
                g.DrawString(x.ToString("#.#"), fn, br, xX(x), yY(hier_xAchse) + 3);
                g.DrawRectangle(pen, xX(x), yY(hier_xAchse), 1, 3);
                x += dx;
            }
            x = -dx;
            while (x > xmin)
            {
                g.DrawString(x.ToString(), fn, br, xX(x), yY(hier_xAchse) + 3);
                g.DrawRectangle(pen, xX(x), yY(hier_xAchse), 1, 3);
                x -= dx;
            }
            x = dy;
            while (x < ymax)
            {
                g.DrawString(x.ToString("0.#####"), fn, br, xX(hier_yAchse) + 5, yY(x));
                g.DrawRectangle(pen, xX(hier_yAchse), yY(x), 3, 1);
                x += dy;
            }
            x = -dy;
            while (x > ymin)
            {
                g.DrawString(x.ToString("0.#####"), fn, br, xX(hier_yAchse) + 5, yY(x));
                g.DrawRectangle(pen, xX(hier_yAchse), yY(x), 3, 1);
                x -= dy;
            }
        }

        private void opencom(object sender, EventArgs e)
        {
            if (serialPort1 != null)
            {
                try
                {
                    serialPort1.Open();
                    this.Text = "Com_x open";
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error_1: " + ex.Message);
                }
                if (serialPort1.IsOpen)
                {
                    serialPort1.DiscardInBuffer();
                    serialPort1.DiscardOutBuffer();
                }
            }
        }
```

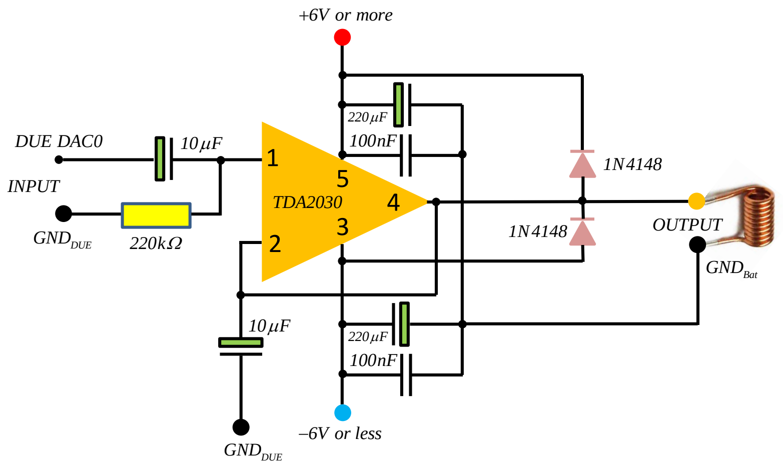
```
        private void closecom(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                try
                {
                    serialPort1.DiscardInBuffer();
                    serialPort1.Close();
                    this.Text = "Com_x closed";
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error_2: " + ex.Message);
                }
            }
        }

        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void drawDUEA1ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            draw_A1 = true;
            sDUE = "A";
            this.BeginInvoke(new EventHandler(opencom));
            this.Invalidate();
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics g = e.Graphics;
            Font fn = new Font("Verdana", 48);
            Brush br = new SolidBrush(Color.Black);
            Brush br_clear = new SolidBrush(Color.AntiqueWhite);
            Pen pen = new Pen(Color.Red, 1/*Strichdicke*/);
            Pen pen0 = new Pen(Color.DarkSeaGreen, 3/*Strichdicke*/);
            g.Clear(Color.AntiqueWhite);
            Koordinatensystem(g, xmax / 5, ymax / 5, xA, yA);
            if (serialPort1.IsOpen && draw_A1)
            {
                Channel_A1(g, Color.Red);
            }
        }
    }
}
```

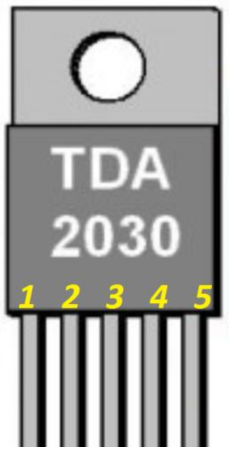
We get a perfect linear triangle wave

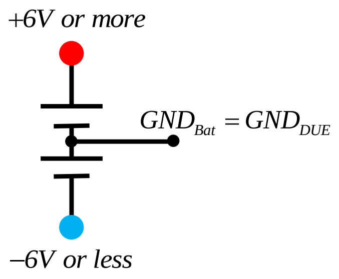

Com_x closed

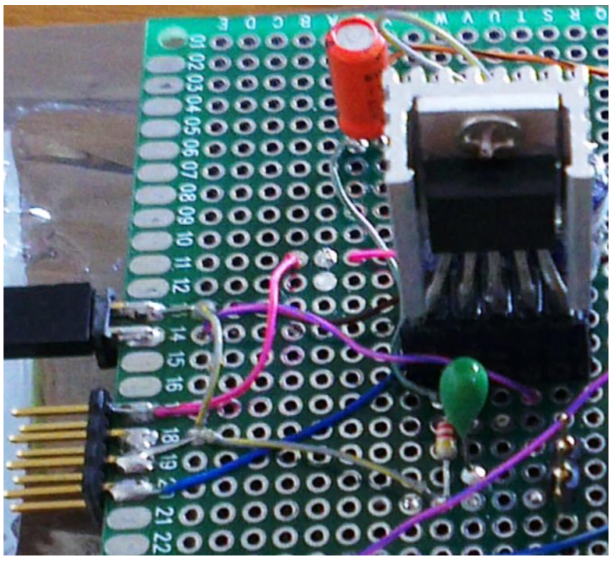
f = 28,894Hz

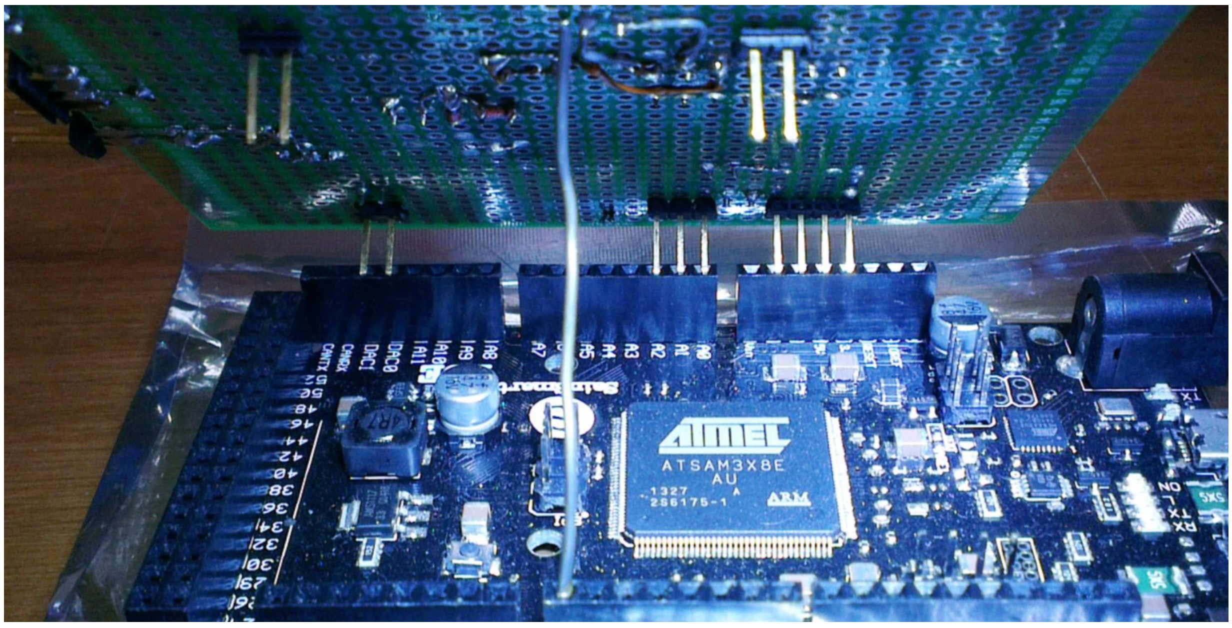
DAC0 is not able to provide the necessary current for coil400 we need. On this ground we design an Arduino DUE shield with a high output current amplifier – TDA2030.











Now we have enough power for coil400 to induce a voltage U_{ind} in coil200. To see what kind of wave function U_{ind} is, we need a second DUE_channel; we choose DUE_pin A0 which is SAM3X8E_channel A7. We upgrade DUE's program by adding some more code …

```
case 'B':
      ADC->ADC_CHER=0xC0;
      delay(100);
      for(n=0; n<500; n++)//for calibration
      {
         analogWrite(DAC0, 0);
         while((ADC->ADC_ISR & 0xC0)!=0xC0); // wait for conversion
         m=ADC->ADC_CDR[7]; // read data at pin A0
         m=ADC->ADC_CDR[6]; // read data at pin A1
      }//not our measurement
      i=0;
      //begin of measurement
      t0 = micros();
      for(n=0; n<4096; n+=dn)
      {
         analogWrite(DAC0, n);
         while((ADC->ADC_ISR & 0xC0)!=0xC0); // wait for conversion
         values[i++]=ADC->ADC_CDR[7]; // read data at pin A0
         values[i++]=ADC->ADC_CDR[6]; // read data at pin A1
      }
      for(n=4095; n>=0; n-=dn)
      {
         analogWrite(DAC0, n);
         while((ADC->ADC_ISR & 0xC0)!=0xC0); // wait for conversion
         values[i++]=ADC->ADC_CDR[7]; // read data at pin A0
         values[i++]=ADC->ADC_CDR[6]; // read data at pin A1
      }
      t1 = micros();
      //end of measurement
      brun = true;
      while(Serial.available() > 0) Serial.read();//clear InBuffer "stop_DUE" from C#
      n=0;
      while(brun && n<(i-1))
      {
         Serial.println(values[n++]);
         Serial.println(values[n++]);
         n+=2*ds;
         if(n>=(i-1)) n=0;
         if(Serial.available() > 0)   brun = false;
      }
      while(Serial.available() > 0) Serial.read();//clear InBuffer
      delay(200);
      Serial.println("begin");
      Serial.println(t1 - t0);
      Serial.println("end");
      break;
```

... and we expand the C# code

```
using System.Collections;

Boolean draw_A0_A1 = false;
ArrayList MWL_a = new ArrayList();


public void Channel_A0_A1(Graphics g, Color c, String do_char) //delay >=1 wählen
        {
            if (serialPort1.IsOpen && draw_A0_A1)
            {
                double x;
                int i, xvon, yvon, xbis, ybis, n_max, mw, ntime = 0;
                Pen pen = new Pen(c, npen);
                Font fn = new Font("Verdana", 48);
```

```
Brush br = new SolidBrush(Color.Black);
String txt;
Boolean b_result;

x = 0;
xvon = xX(x); n_max = this.ClientRectangle.Width;
wait_ms(100);
serialPort1.ReadExisting();
wait_ms(100);
serialPort1.Write(do_char);
wait_ms(200);
b_result = int.TryParse(serialPort1.ReadLine(), out mw);
if (b_result)
    MWL_a.Add(mw);
else
    MWL_a.Add(0);
b_result = int.TryParse(serialPort1.ReadLine(), out mw);
if (b_result)
    yvon = yY(mw);
else
    yvon = yY(0);
xbis = xvon + dn;
while (xvon < n_max)
{
    b_result = int.TryParse(serialPort1.ReadLine(), out mw);
    if (b_result)
        MWL_a.Add(mw);
    else
        MWL_a.Add(0);
    b_result = int.TryParse(serialPort1.ReadLine(), out mw);
    if (b_result)
        ybis = yY(mw);
    else
        ybis = yY(0);
    g.DrawLine(pen, xvon, yvon, xbis, ybis);
    xvon = xbis; yvon = ybis;
    xbis = xvon + dn;
}
pen = new Pen(Color.Blue, npen);
i = 0; x = 0; xvon = xX(x); yvon = yY((int)MWL_a[i++]);
xbis = xvon + dn;
while (xvon < n_max)
{
    ybis = yY((int)MWL_a[i++]);
    g.DrawLine(pen, xvon, yvon, xbis, ybis);
    xvon = xbis; yvon = ybis;
    xbis = xvon + dn;
}
MWL_a.Clear();
//--------------------Time-------------
serialPort1.Write("z");
serialPort1.ReadExisting();
wait_ms(100);
serialPort1.ReadExisting();
wait_ms(100);
txt = "";
for (i = 0; i < 3; i++)
    txt += serialPort1.ReadLine();
//g.DrawString(txt, fn, br, 50, 150);
if (txt.Contains("begin") && txt.Contains("end"))
{
    String snum = txt.Remove(0, 5);
    snum = snum.Remove(snum.IndexOf("end"));
    b_result = int.TryParse(snum, out ntime);
}
```

```
                if (b_result) g.DrawString("f = " + (1000000.0 / ntime).ToString("0.000") +
"Hz", fn, br, 50, 50);

                this.BeginInvoke(new EventHandler(closecom));
            }
            draw_A0_A1 = false;
        }
```

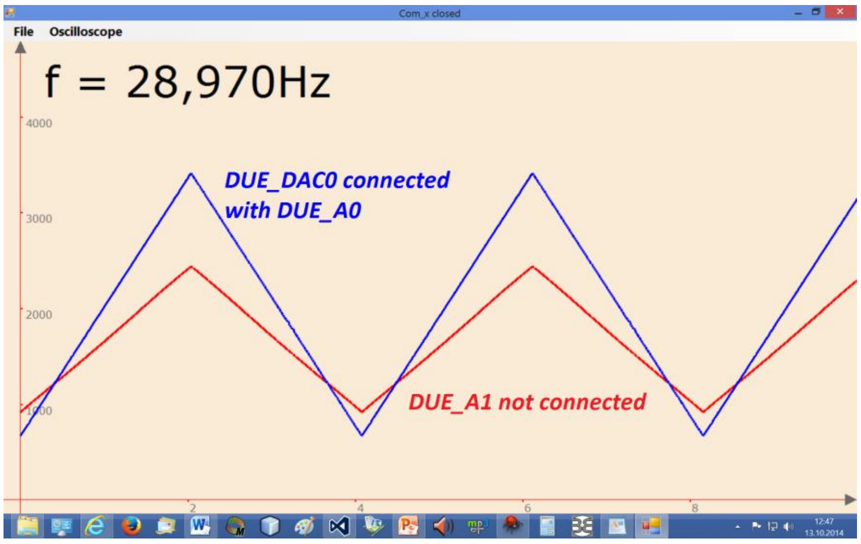
Menue

```
private void drawDUEA0A1ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            draw_A0_A1 = true;
            sDUE = "B";
            this.BeginInvoke(new EventHandler(opencom));
            this.Invalidate();
        }
```

Inside paint

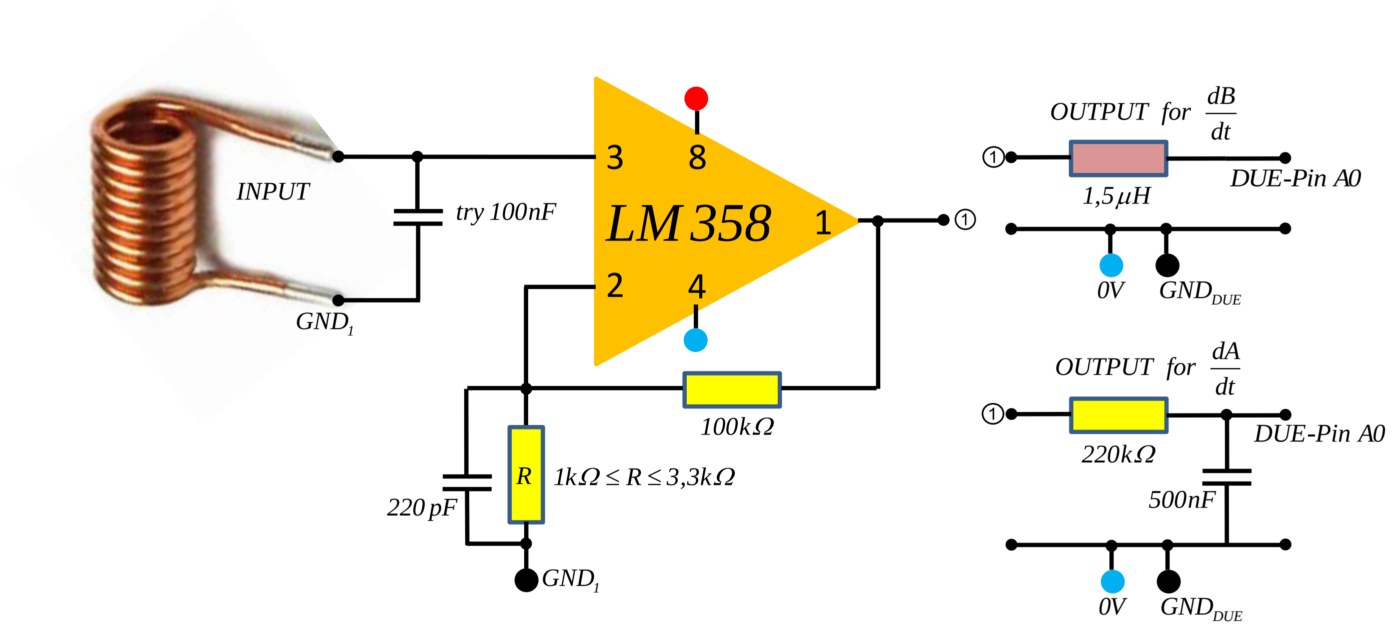
```
if (serialPort1.IsOpen && draw_A0_A1)
            {
                Channel_A0_A1(g, Color.Red, sDUE);
            }
```

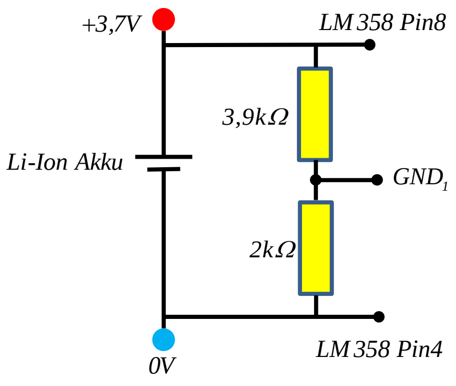
At first we don't use the shield but a wire to connect DUE_DAC0 with DUE_A0; DUE_A1 is open. We get

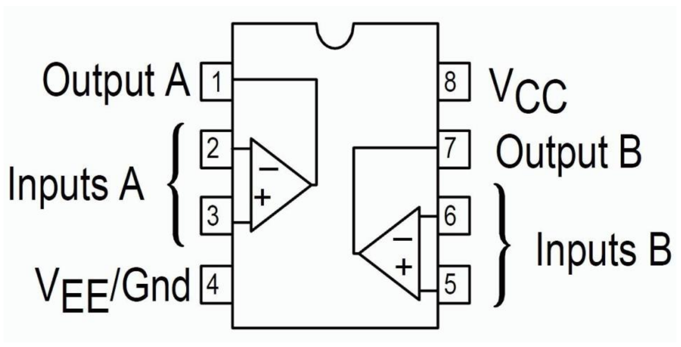


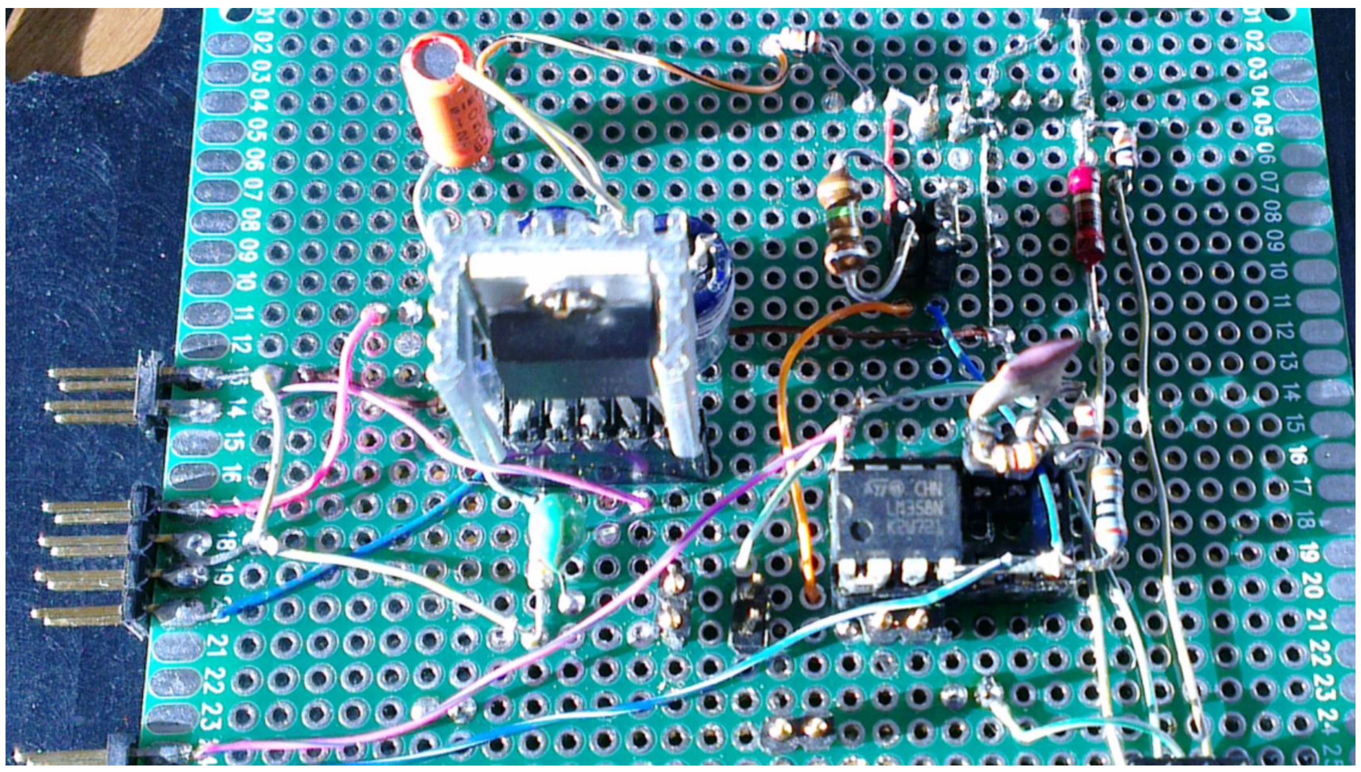
Obviously open DUE_A1 channel is following channel DUE_A0, which has a defined signal.

From coil200 we receive a week AC voltage, not suited for channel DUE_A0. It has to be process electronically before send to DUE_A0. For this we add a low power supply operational amplifier LM358 to the self-made Arduino DUE shield. Its job is to amplify and to shift the U_{ind} signal in a way that the output voltage of LM358 is high enough in magnitude and never negative.

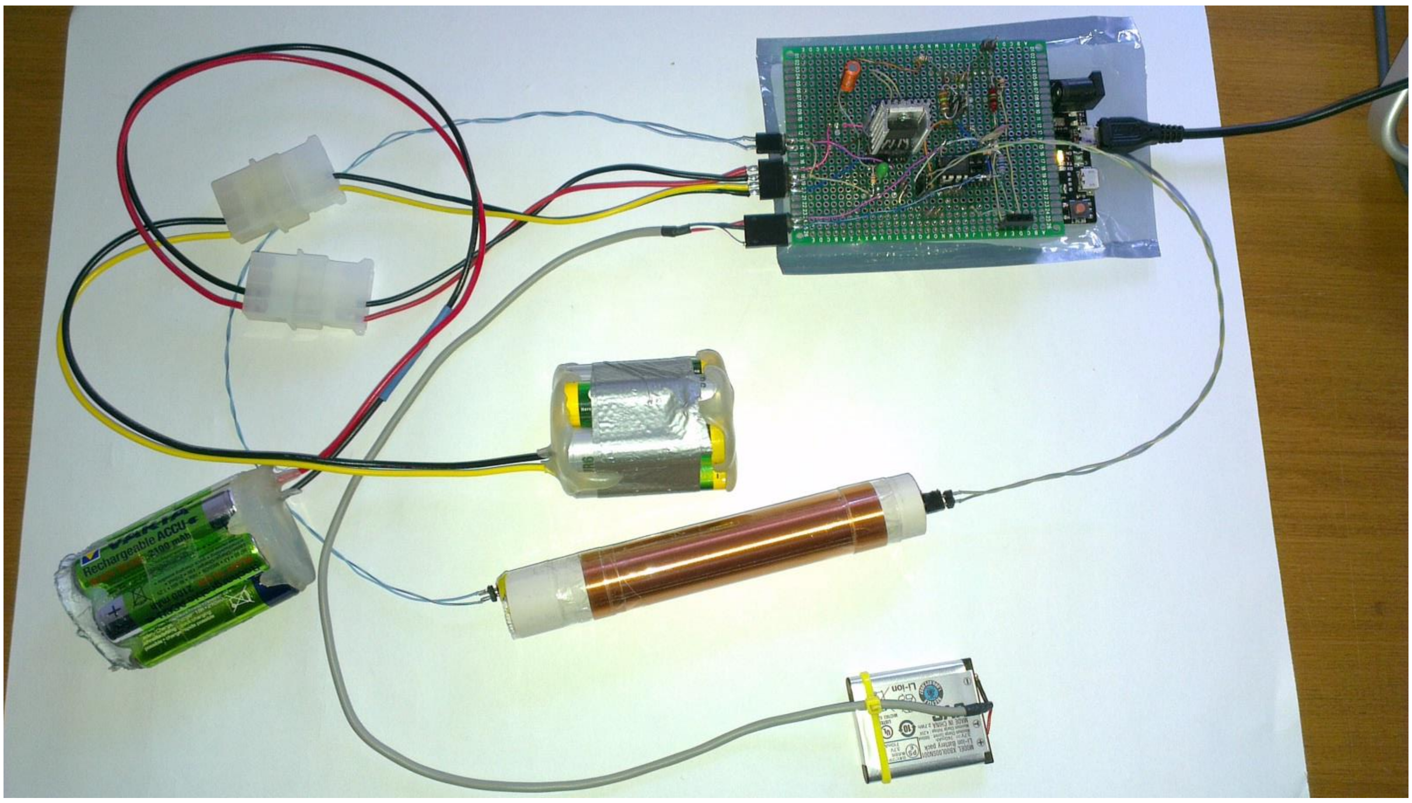


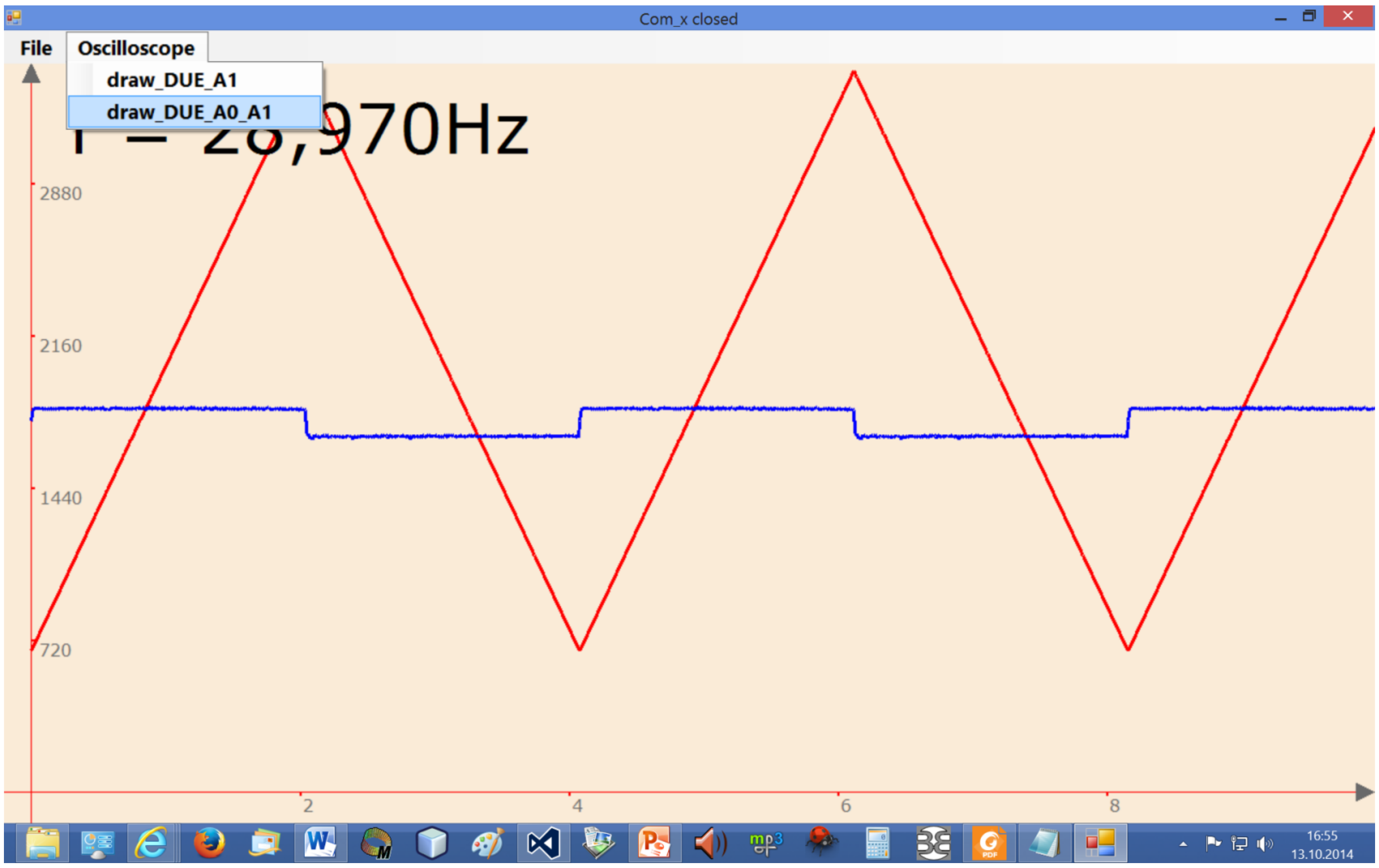






If we assemble all components we end up with:





For measurements we need different slopes of the ramp of our perfect triangle wave. We achieve this by different steps inside the for-loop:

`for(n=0; n<4096; n+=dn) { analogWrite(DAC0, n); ...` *We choose* $1 \le dn \le 10$ *for up and down.*

To avoid densely packed signals of the C# oscilloscope we use different ds steps in addition.

```
while(brun && n<i) { Serial.println(values[n]); n+=ds; ...
```

To enter all the dn numbers required, we introduce a second form in C# for a dialog.

For Arduino DUE the software upgrade is:

```
      case 'C':
      while(!(Serial.available() > 0)); // wait for SendData
      stxt = ""; c1 = Serial.read(); stxt += c1;
      dn = stxt.toInt();
      if((dn<1) || (dn>10)) dn = 10;
      Serial.println(dn);//response
      switch(dn)
      {
         case 10: ds=1; break;
         case 9: case 8: ds=2; break;
         case 7: case 6: ds=3; break;
         case 5: case 4: ds=4; break;
         case 3: ds=6; break;
         case 2: ds=8; break;
         case 1: ds=15; break;
      }
      while (Serial.available()>0) Serial.read();
      break;
```

For C# we have to write a little bit more:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TriWave
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        public void set_df(String txt) { this.textBox1.Text = txt; }
        public void set_dn(String txt) { this.textBox2.Text = txt; }

        public int get_df()
        {
            int num;
            bool res = int.TryParse(this.textBox1.Text, out num);
            if (res == false) return 10;
            else return num;
        }
        public int get_dn()
        {
            int num;
            bool res = int.TryParse(this.textBox2.Text, out num);
            if (res == false) return 1;
            else return num;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.Ignore;
            this.Close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
        }
    }
}
```

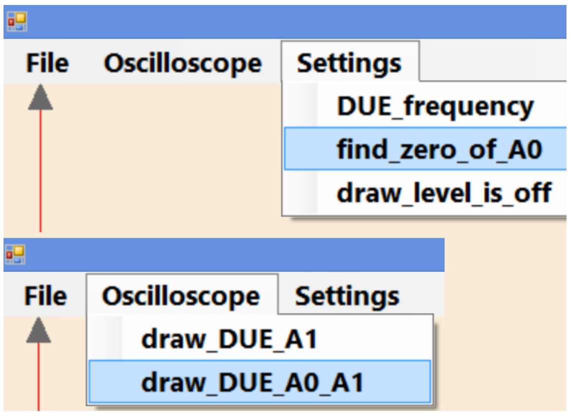
Inside Form1 insert:

```
Boolean get_zero_of_A0 = false, draw_level_line = false;
Point pt1;

private void dUEfrequencyToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form2 Dlg1 = new Form2();
            Dlg1.set_df(df.ToString());
            Dlg1.set_dn(dn.ToString());
            if (Dlg1.ShowDialog(this).Equals(DialogResult.OK))
            {
                df = Dlg1.get_df();
                dn = Dlg1.get_dn();
                if ((df < 1) || (df > 9)) df = 10;
                this.Refresh();
                serialPort1.Open();
                wait_ms(100);
                serialPort1.ReadExisting();
                wait_ms(100);
                serialPort1.ReadExisting();
                wait_ms(100);
                serialPort1.Write("C");
                wait_ms(100);
                if (df == 10) serialPort1.Write("0");
                else serialPort1.Write(df.ToString());
                wait_ms(100);
                String received = serialPort1.ReadLine();
                serialPort1.Close();
                MessageBox.Show("DUE received:\ndf = " + received, "DUE's Response");
            }
            else
            {
                //System.Windows.Forms.MessageBox.Show("Dlg1");
            }
            Dlg1.Close();
        }

        private void findzeroofA0ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            get_zero_of_A0 = true;
        }

        private void drawlevelisoffToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (draw_level_line)
            {
                draw_level_line = false;
                this.drawlevelisoffToolStripMenuItem.Text = "draw_level_is_off";
            }
            else
            {
                draw_level_line = true;
                this.drawlevelisoffToolStripMenuItem.Text = "draw_level_is_on";
            }
        }

        private void Form1_MouseUp(object sender, MouseEventArgs e)
        {
            if (draw_level_line && e.Button == MouseButtons.Left)
            {
                pt1 = new Point(e.X, e.Y);
                Graphics g = this.CreateGraphics();
                Font fn = new Font("Verdana", 48);
                Brush br = new SolidBrush(Color.Black);
                Pen pen = new Pen(Color.DarkGreen, 3/*Strichdicke*/);
                xStreckfaktor = this.ClientRectangle.Width / (xmax - xmin);
                yStreckfaktor = this.ClientRectangle.Height / (ymax - ymin);
                g.DrawLine(pen, xX(xmin), pt1.Y, xX(xmax), pt1.Y);
                g.DrawString(((int)Yy(pt1.Y)).ToString(), fn, br, pt1.X, pt1.Y - 70);
            }
        }
```

Inside `public void Channel_A0_A1(Graphics g, ...` *look for* `MWL_a.Clear();` *and insert:*

```
                ...
                if (get_zero_of_A0)
                {
                    int sum = 0, n = 0;
                    for (i = (MWL_a.Count / 2); i < MWL_a.Count; i++ )
                    {
                        sum += (int)MWL_a[i]; n++;
                    }
                    hier_0 = sum / n;
                    get_zero_of_A0 = false;
                }
                MWL_a.Clear();
```

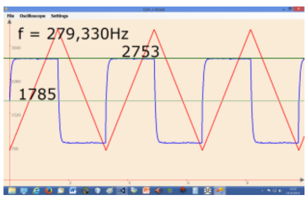
Now we start a bunch of measurements to find out $U_{ind}\left(\frac{dB}{dt}\right) \sim U_{ind}\left(\frac{dI}{dt}\right) \underset{\text{this triangle wave}}{\sim} U_{ind}(f)$.

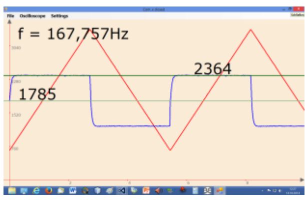


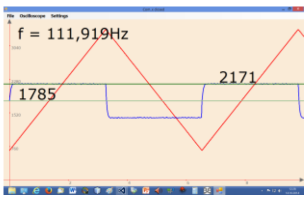
*Zero line of channel A0: Before we connect coil200 to the input of LM358 link over the input pins with a jumper choose **find_zero_of_A0** and start **draw_DUE_A0_A1**. Replace the jumper by coil200.*

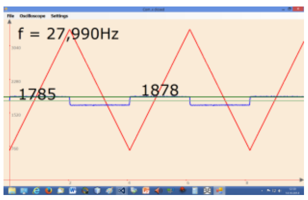
*Choose **DUE_frequency** and complete the dialog with df=10 and dn=2 (or dn=1), then start **draw_DUE_A0_A1**. Choose **draw_level_is_off**; it changes to **draw_level_is_on**, and use a LMB click to draw a line through the maximum of the blue square wave. Repeat this step for df=9 until df=1.*

Some of ten measurements:



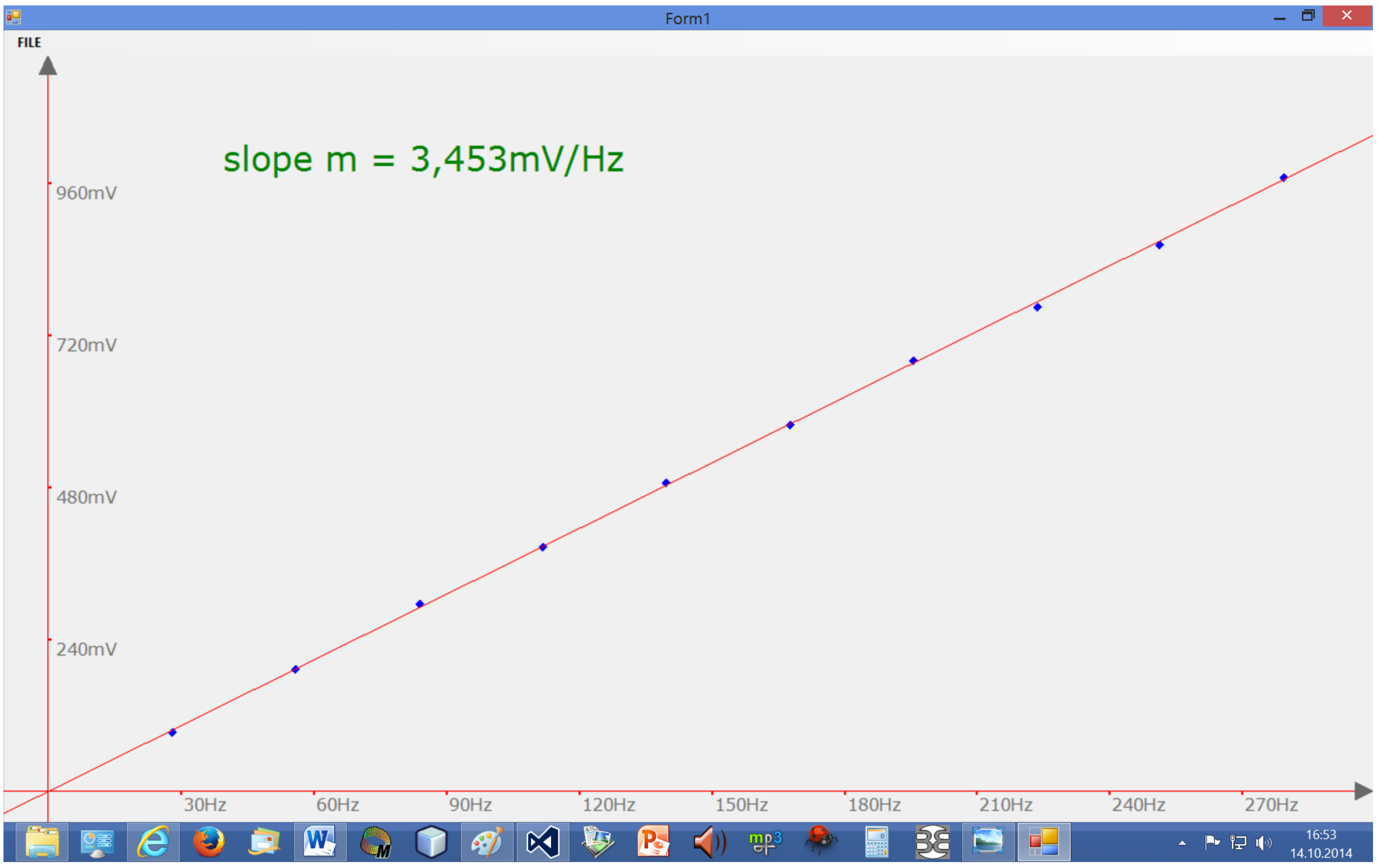






A mathematical evaluation together with the method of least squares for fitting the curve in C#:

```
double zero_line = 1785;
double[] Sp_U = { 1878,1979,2081,2171,2273,2364,2465,2550,2647,2753 };//voltage U
double[] Frq_f = { 27.99,55.98,83.91,111.92,139.74,167.76,195.5,223.71,251.26,279.33 };//frequency


public void draw_fct(Graphics g, TFkt_00 z, Color c)
        {
            double x, dx;
            int xvon, yvon, xbis, ybis;
            Pen pen = new Pen(c, 1);
            dx = (xmax - xmin) / this.ClientRectangle.Width;
            x = xmin; xvon = xX(x); yvon = yY(z.f(x));
            x += dx;
            while (x <= xmax)
            {
                xbis = xX(x); ybis = yY(z.f(x));
                g.DrawLine(pen, xvon, yvon, xbis, ybis);
                x += dx; xvon = xbis; yvon = ybis;
            }
        }

public void draw_pt(Graphics g, Color c)
        {
            int xvon, yvon, i = 0, d = 2;
            Pen pen = new Pen(c, 3 * d);
            while (i < Sp_U.Length)
            {
                xvon = xX(Frq_f[i]); yvon = yY(Sp_U[i] - zero_line);
                g.DrawLine(pen, xvon - d, yvon - d, xvon + d, yvon + d);
                i++;
            }
        }

        public double lsqfit()
        {
            double z = 0, n = 0;
            for (int i = 0; i < Sp_U.Length; i++)
            {
                z += Frq_f[i] * (Sp_U[i] - zero_line);
                n += Frq_f[i] * Frq_f[i];
            }
            return z / n;
        }

private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics g = e.Graphics;
            xStreckfaktor = this.ClientRectangle.Width / (xmax - xmin);
            yStreckfaktor = this.ClientRectangle.Height / (ymax - ymin);
            Koordinatensystem(g, Color.Azure, xmax / 10, ymax / 5);
            draw_pt(g, Color.Blue);
            double D = lsqfit();
            TFkt_02 h02 = new TFkt_02(D);
            draw_fct(g, h02, Color.Red);
            Font fn = new Font("Verdana", 24);
            Brush br = new SolidBrush(Color.Green);
            g.DrawString("slope m = " + D.ToString("0.####") + "mV/Hz", fn, br, 200, 100);
}

public class TFkt_00 { public virtual double f(double x) { return x; } } //basic class
public class TFkt_02 : TFkt_00
{//1. extension
    double m0;
    public TFkt_02(double m)
    {
        m0 = m;
    }
    public override double f(double x) { return m0 * x; }
}
```



… and what if coil400 receives a current in form of a parabola function or sinus function. For Arduino DUE not a problem: if coil400 produce a magnetic field $B(t) \sim I(t)$ *then the induced EMF of coil200 is proportional to the derivative of I(t):* $U_{ind}(t) \sim I'(t)$.

***Parabola example**: DUE's part*

```
uint16_t values[24992], u16_fct[820];
```

Add inside switch:

```
case 'L': //Parabola
      ADC->ADC_CHER=0xC0;
      delay(100);
      m = 0;
      for(n=0; n<820; n++)
      {   //values of a parabola open downwards
         u16_fct[m++] = (uint16_t)(4094.0 - 4094.0/168100.0*(n-410.0)*(n-410.0));
      }
      for(n=0; n<820; n++)//for calibration
      {
         analogWrite(DAC0, u16_fct[n]);
         while((ADC->ADC_ISR & 0xC0)!=0xC0); // wait for conversion
         m=ADC->ADC_CDR[7]; // read data at pin A0
         m=ADC->ADC_CDR[6]; // read data at pin A1
      }//not our measurement
      i=0;
      //begin of measurement
      t0 = micros();
      for(n=0; n<820; n++)
      {
         analogWrite(DAC0, u16_fct[n]); // use DAC1 for ... DAC1
         while((ADC->ADC_ISR & 0xC0)!=0xC0); // wait for conversion
         values[i++]=ADC->ADC_CDR[7]; // read data at pin A0
         values[i++]=ADC->ADC_CDR[6]; // read data at pin A1
      }
      t1 = micros();
      //end of measurement
      brun = true;
      while(Serial.available() > 0) Serial.read();//clear InBuffer "stop_DUE" from C#
      n=0;
      while(brun && n<(i-1))
      {
         Serial.println(values[n++]);
         Serial.println(values[n++]);
         n+=2*ds;
         if(n>=(i-1)) n=0;
         if(Serial.available() > 0)   brun = false;
      }
      while(Serial.available() > 0) Serial.read();//clear InBuffer
      delay(200);
      Serial.println("begin");
      Serial.println(t1 - t0);
      Serial.println("end");
      break;
```

and C# code for the oscilloscope

```
Boolean draw_parabola = false;

private void drawparaboladownToolStripMenuItem_Click(object sender, EventArgs e)
        {
            draw_parabola = true;
            sDUE = "L";
            this.BeginInvoke(new EventHandler(opencom));
            this.Invalidate();
        }
```

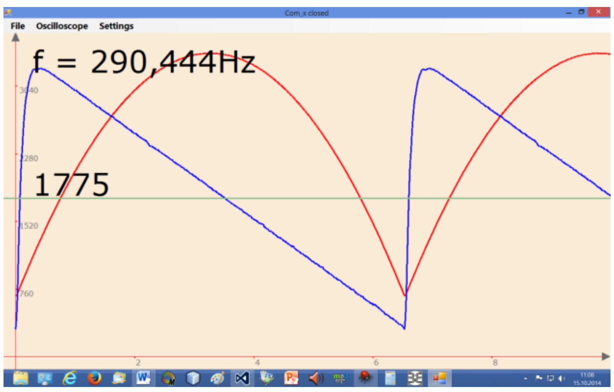
... and a supplement of paint:

```
if (serialPort1.IsOpen && draw_parabola)
            {
                Channel_A0_A1(g, Color.Red, sDUE);
                g.DrawLine(pen0, xX(xmin), yY(hier_0), xX(xmax), yY(hier_0));
                g.DrawString(hier_0.ToString(), fn, br, 50, yY(hier_0) - 70);
            }
```

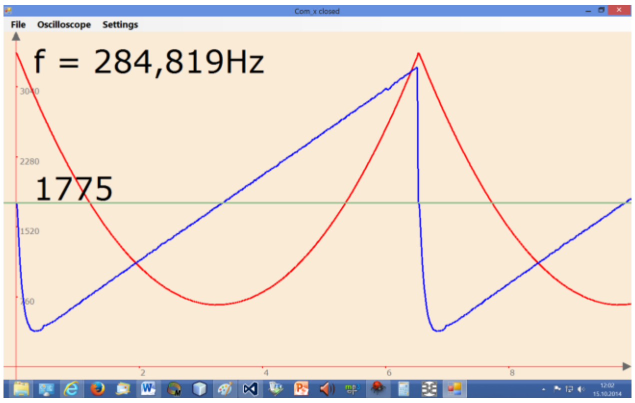
$U_{ind}(t)$ is now a falling line because

$$I(t) = -t^2 \quad \Rightarrow \quad I'(t) = -2 \cdot t$$

C# dialog with dn=2.

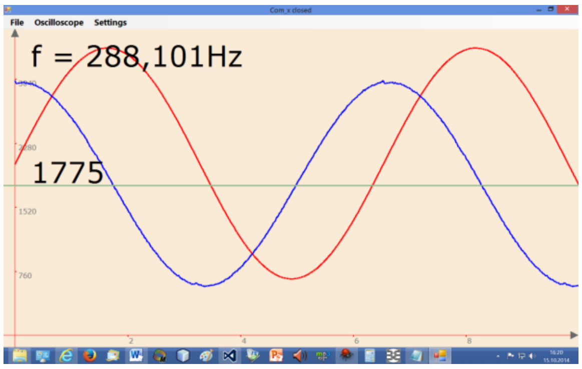


Task: To show that for a parabola current I(t) of coil400, opens upwards, a (your) result of $U_{ind}(t)$ of coil200 is as alongside.



Sinus example: *C# part is easy, therefore only DUE's part*

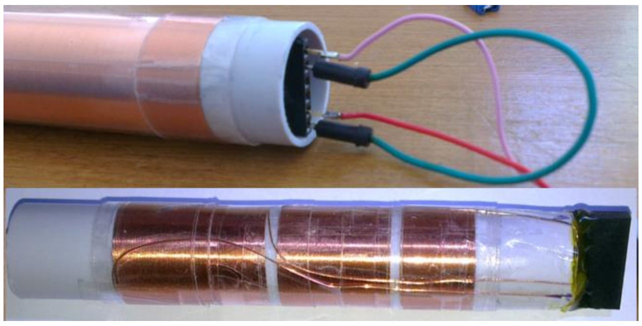
```
case 'E': //Sinus
      ADC->ADC_CHER=0xC0;
      delay(100);
      m = 0;
      for(n=0; n<820; n++)
      {
         u16_fct[m++] = (uint16_t)(2047 +2047*sin(2*PI/820.0*n));
      }
      for(n=700; n<820; n++)//for calibration
      {
         analogWrite(DAC0, u16_fct[n]);
         while((ADC->ADC_ISR & 0xC0)!=0xC0); // wait for conversion
         m=ADC->ADC_CDR[7]; // read data at pin A0
         m=ADC->ADC_CDR[6]; // read data at pin A1
      }//not our measurement
      i=0;
      t0 = micros();//begin of measurement
      for(n=0; n<820; n++)
      {
         analogWrite(DAC0, u16_fct[n]);
         while((ADC->ADC_ISR & 0xC0)!=0xC0); // wait for conversion
         values[i++]=ADC->ADC_CDR[7]; // read data at pin A0
         values[i++]=ADC->ADC_CDR[6]; // read data at pin A1
      }
      t1 = micros();//end of measurement
      brun = true;
      while(Serial.available() > 0) Serial.read();//clear InBuffer "stop_DUE" from C#
      n=0;
      while(brun && n<(i-1))
      {
         Serial.println(values[n++]);
         Serial.println(values[n++]);
         n+=2*ds;
         if(n>=(i-1)) n=0;
         if(Serial.available() > 0)   brun = false;
      }
      while(Serial.available() > 0) Serial.read();
      delay(200);
      Serial.println("begin");
      Serial.println(t1 - t0);
      Serial.println("end");
      break;
```



This should be the end of part ② $U_{ind} = U_{ind}\left(\frac{dB}{dt}\right);\ n = const;\ A = const.$

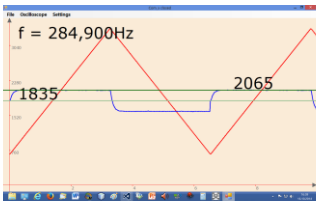
We continue further experiments with gadget DUE with point

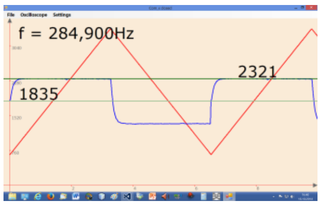
① $U_{ind} = U_{ind}(n);\ \dfrac{dB}{dt} = const;\ \ A = const.$

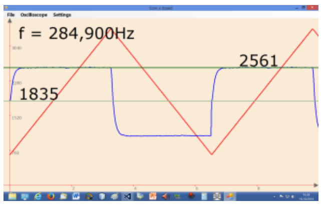


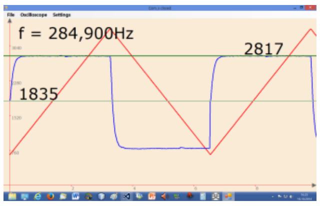
This is a point to relax a little bit; we only wiring up another coil named coil100. This coil100 consists of three parts: 50, 75 and 100 windings and we combine them and together with coil200 we get: 50, 75, 100, 125, 150, 175, 200 and 225 windings to use for a second bunch of induction experiments. Again we use Arduino DUE's ability to generate triangle waves and we look for U_{ind} *by different windings:* $U_{ind}(n)$ *– without writing new code!*

Some of eight measurements:



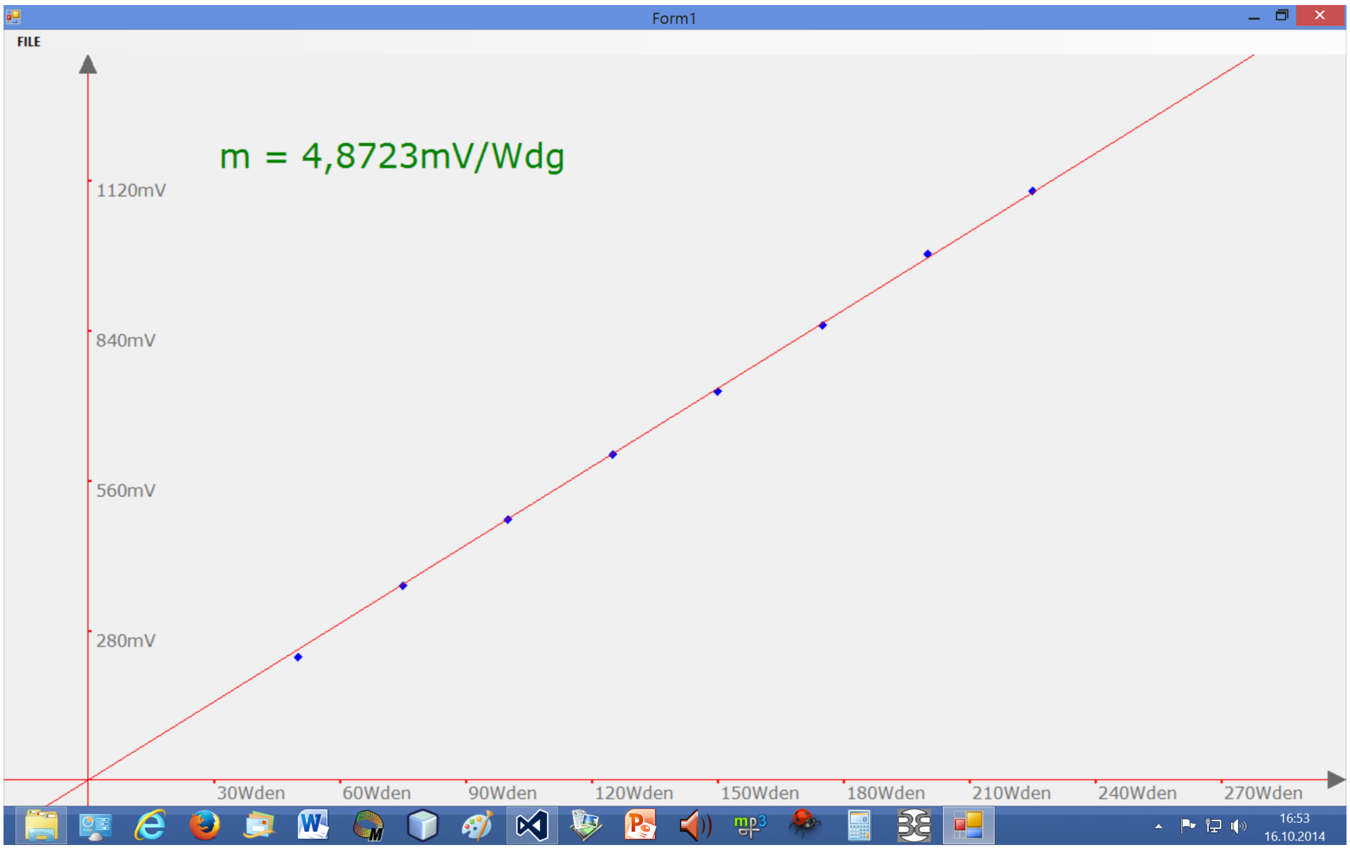






Roughly 50 100 150 200 windings.

A mathematical evaluation together with the method of least squares for fitting the curve in C# leads us to:



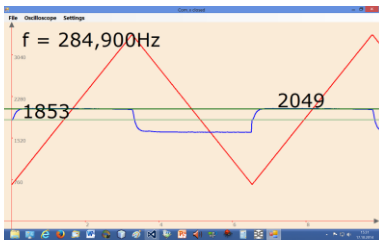
We continue with point ③ $U_{ind} = U_{ind}(A);\; n = const;\; \frac{dB}{dt} = const.$

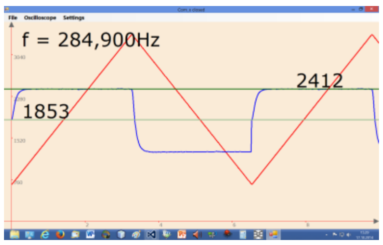
Again no new more code is needed but a set of coils with different cross sectional areas. Let's use circle areas. All the new coils should have 200 windings.

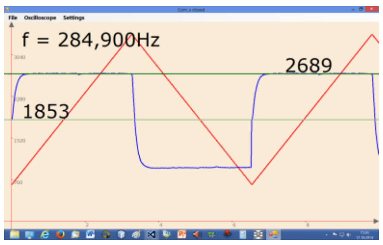
With a caliper we get all the external diameter of the used plastic pipes carrying the windings:

$\varnothing \quad d_1 = 4{,}6mm;\quad d_2 = 5{,}9mm;\quad d_3 = 7{,}5mm;\quad d_4 = 9{,}4mm;\quad d_5 = 10{,}2mm;\quad d_6 = 11{,}4mm;$
$d_7 = 15mm \quad roughly.$

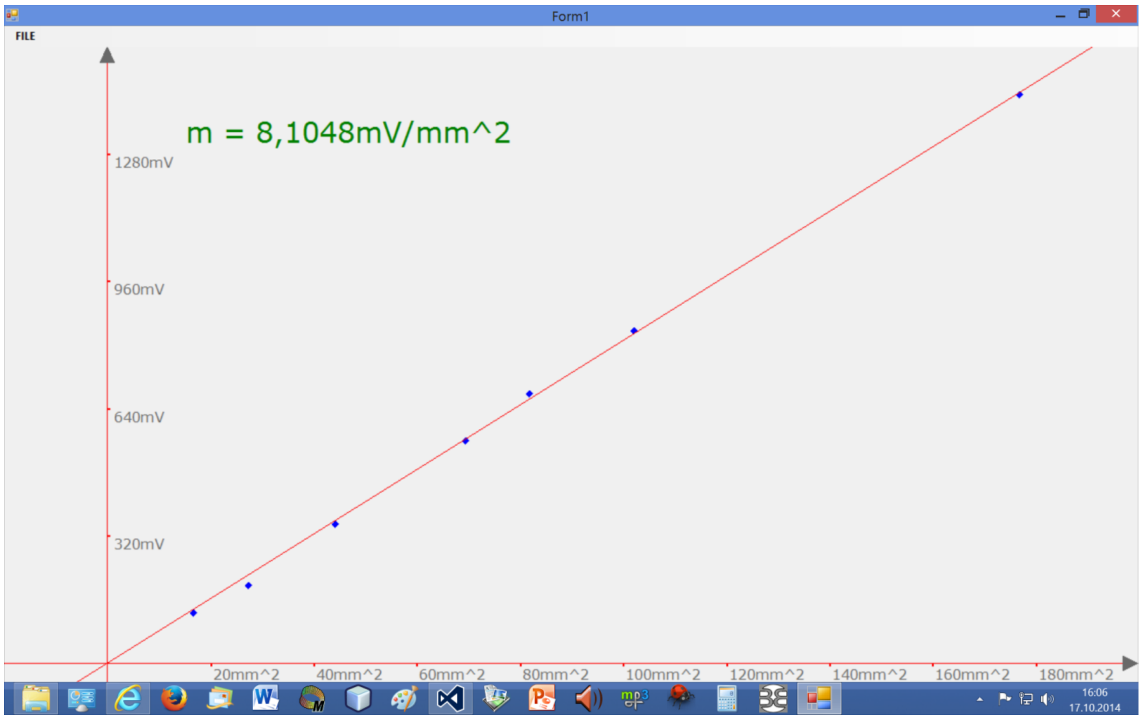
Some of seven measurements:





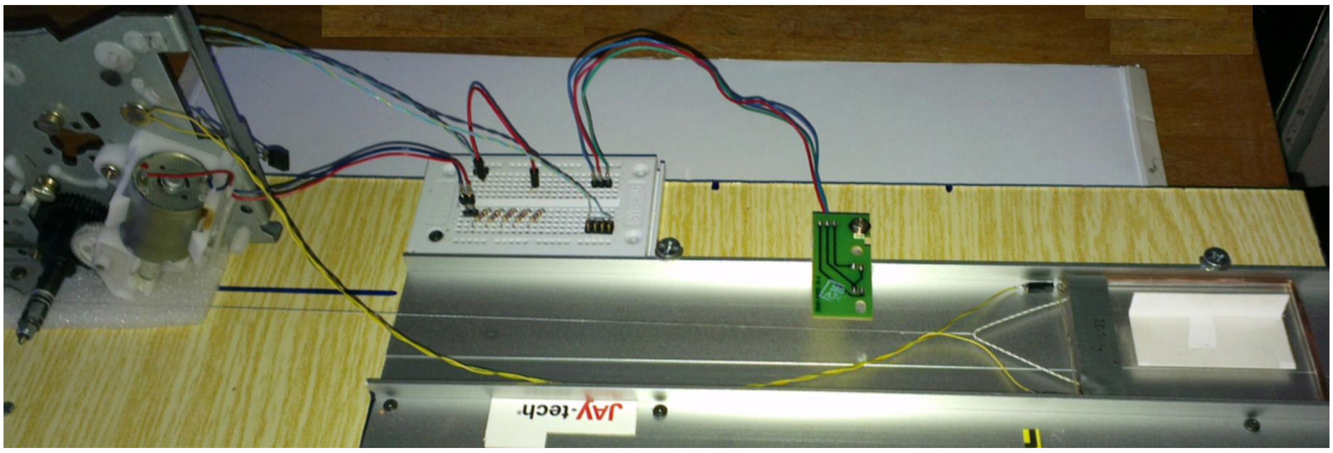


A graphical result is given by the picture hereafter:



Finally we see about ④ $U_{ind} = U_{ind}\left(\frac{dA}{dt}\right)$; $n = const$; $B = const$.

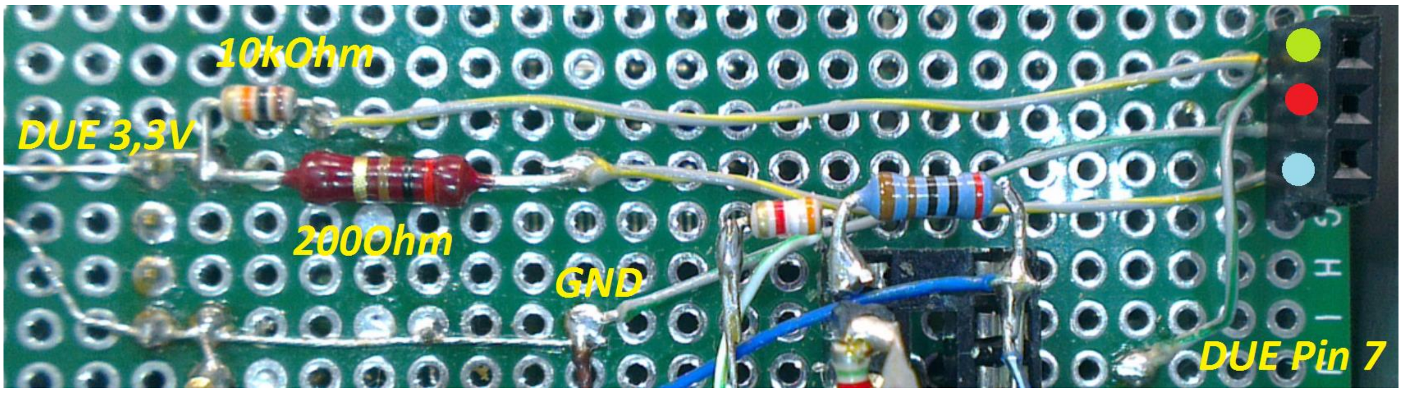
In an experiment we move a rectangular coil of 500 windings slowly by a motor over a constant homogeneous magnetic field. The field is given by a set of neodyn magnets and is not really homogeneous, but still usable for us.



The motor for slowly speeds was part of an old video recorder I disassembled and the small green board is a light barrier for measuring the velocity of the moved coil.

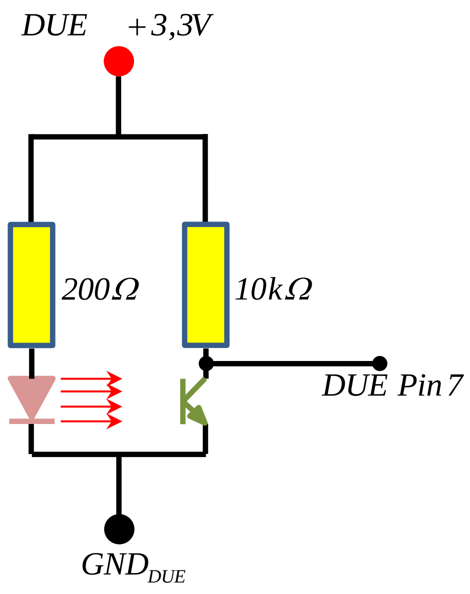
But who measures the velocities $v = \frac{\Delta s}{\Delta t}$*?   Arduino DUE of course via DUE_Pin_7 for time* Δt *!*

We use a strip of paper of length $\Delta s = 5cm$ *on the back of the moved coil to interrupt the beam of the light barrier.*

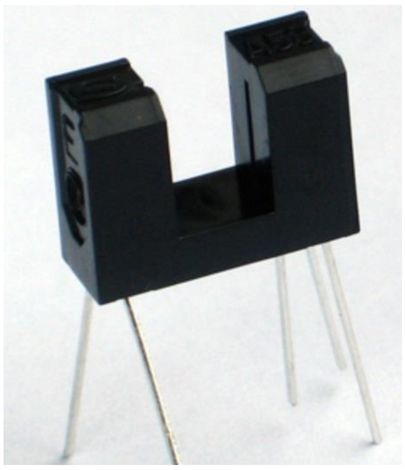


This picture shows our last expansion of the DUE-shield with two resistors to run the light barrier

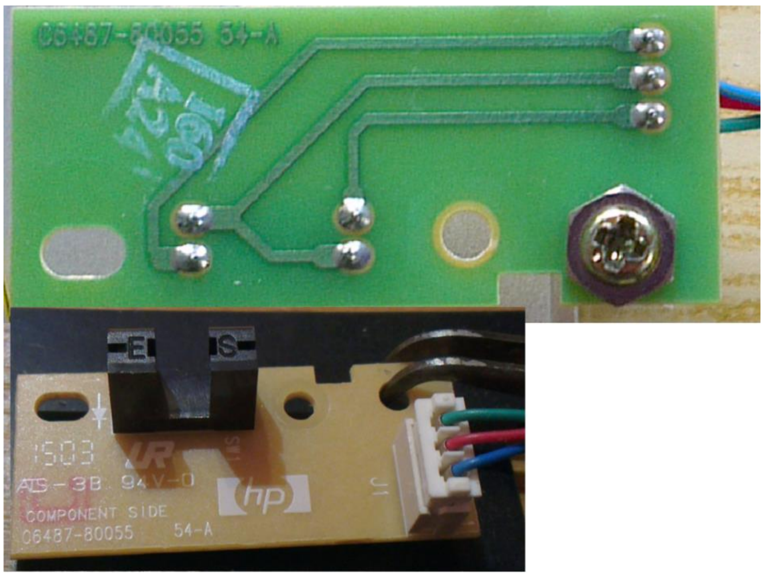
The circuit to run the light barrier



Light Barrier



We use a second-hand circuit we found.



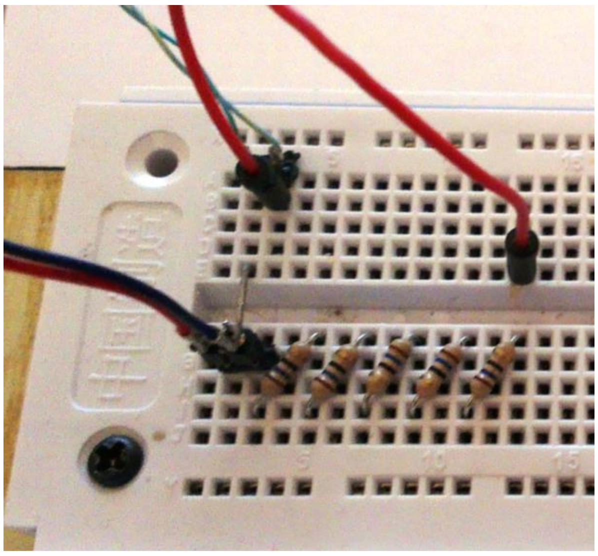
Again we upgrade DUE's program by adding some more code together with an interrupt routine for Δt *.*
The average of 10000 measurements `values[0] = (uint16_t)( m / 10000.0);` *and* `delay(ndel-10);` *are corresponding. Without any average use:* `delay(ndel);`

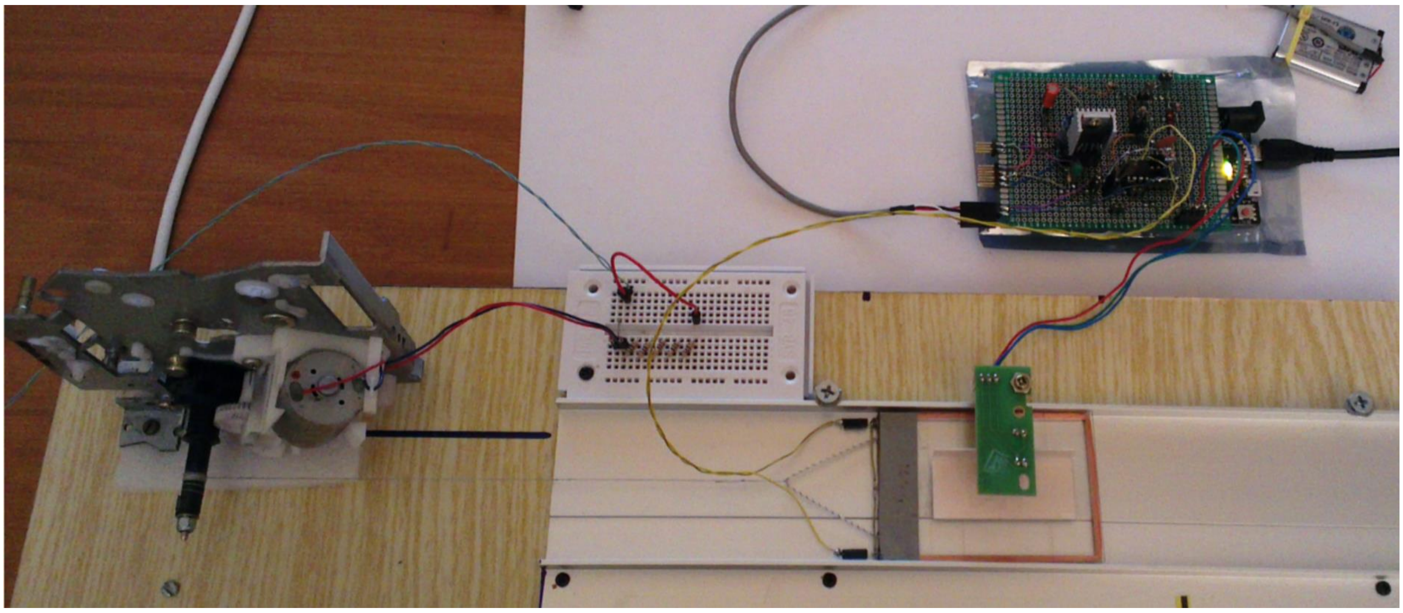
```
int ndel = 15;
const int LB_Pin = 7;//LB: Light_Barrier
String stxt = "";
volatile unsigned long t1 = 123456, t0 = 0;
volatile int step = 1;

void setup()
{
   attachInterrupt(LB_Pin, t_measure, CHANGE);//LB: Light_Barrier
   …
void loop()
{

   …
   //begin of moving coil
      case 'F': step = 1; ndel = 20; read_send(); break;
      case 'G': step = 1; ndel = 19; read_send(); break;
      case 'H': step = 1; ndel = 18; read_send(); break;
      case 'I': step = 1; ndel = 17; read_send(); break;
      case 'J': step = 1; ndel = 16; read_send(); break;
      case 'K': step = 1; ndel = 15; read_send(); break;
      //end of moving coil
   }
}

void read_send()//moving coil
{
   ADC->ADC_CHER=0x80;
   delay(100);
   for(n=0; n<500; n++)//for calibration
   {
      while((ADC->ADC_ISR & 0x80)!=0x80); // wait for conversion
      i=ADC->ADC_CDR[7]; // read data at DUE_pin_A0
   }//not our measurement
   brun = true;
   while(Serial.available() > 0) Serial.read();//clear InBuffer
   while(brun)
   {
      m = 0;
      for(n=0; n<10000; n++)
      {
         while((ADC->ADC_ISR & 0x80)!=0x80);//wait for conversion
         m += ADC->ADC_CDR[7];
      }
      values[0] = (uint16_t)( m / 10000.0); //average
      delay(ndel-10);
      Serial.println(values[0]);
      if(Serial.available() > 0) brun = false; // wait for SendData
   }
   while(Serial.available() > 0) Serial.read();//clear InBuffer
   delay(200);
   Serial.println("begin");
   Serial.println(t1 - t0);
   Serial.println("end");
}

void t_measure()
{
   switch(step)
   {
      case 1:
      //begin of measurement
      t0 = micros();
      step++;
      break;
      case 2:
      //end of measurement
      t1 = micros();
      if((t1-t0)>100000) step++;
      break;
   }
}
```

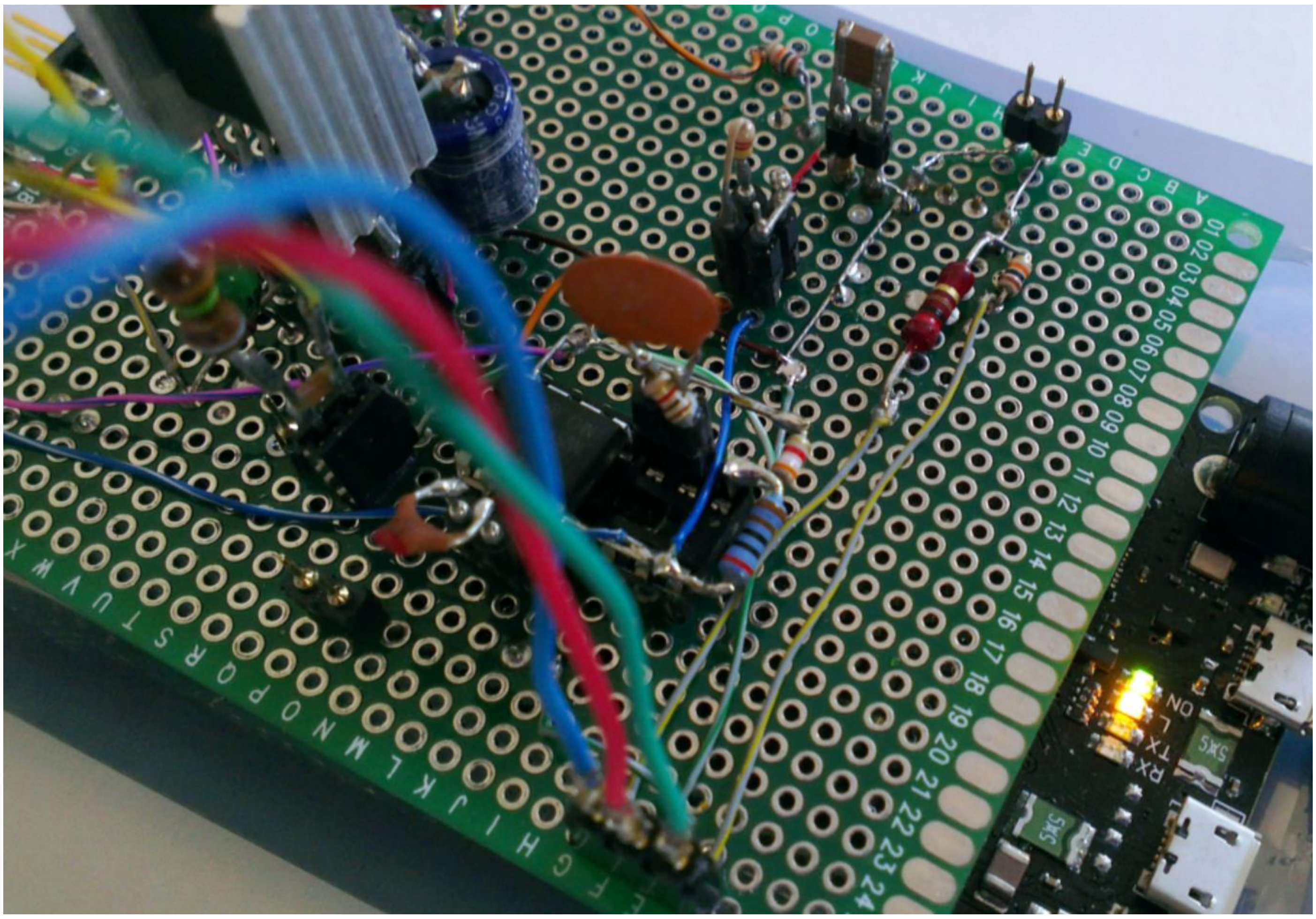
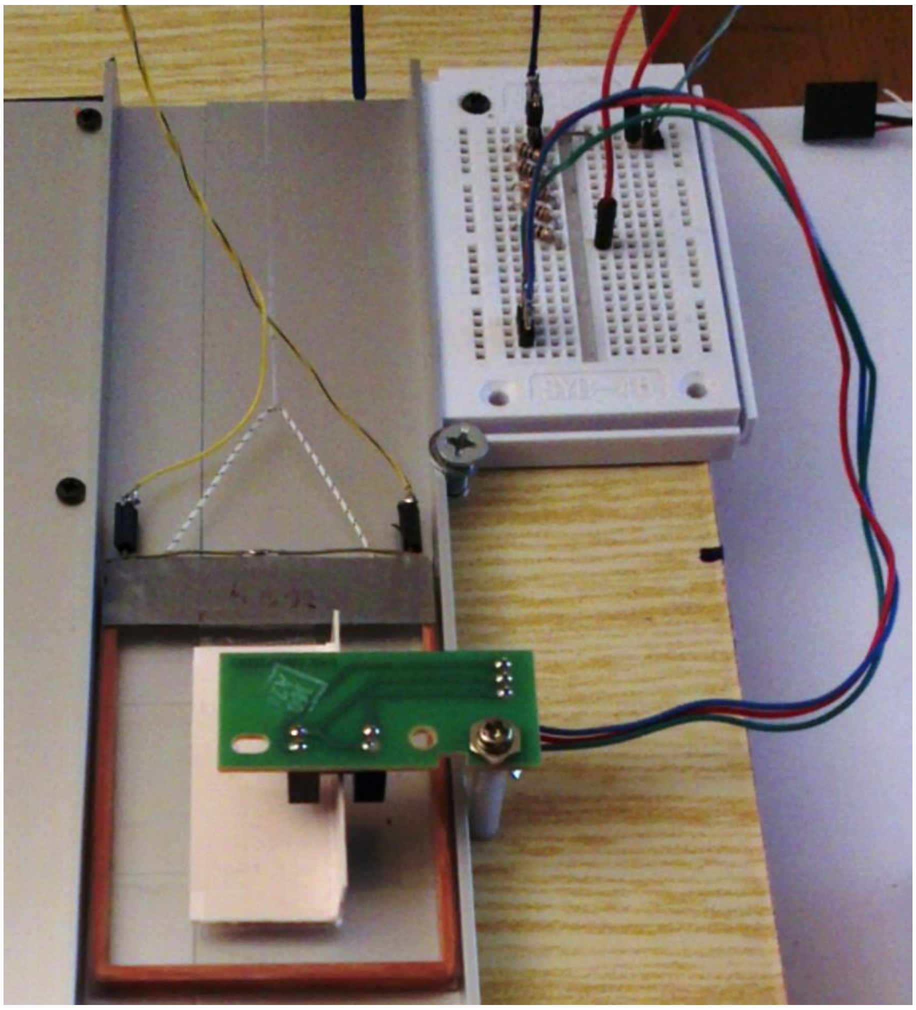
The induction voltage of the moving coil is connected to DUE_pin_A0; therefore in C# we add analog to `public void Channel_A1(Graphics g, Color c)` *a procedure* `public void Channel_A0(Graphics g, Color c, String do_char)` *to draw the induction voltage signals.*

```
public void Channel_A0(Graphics g, Color c, String do_char)
        {
            if (serialPort1.IsOpen && draw_A0)
            {
                double x;
                int i, xvon, yvon, xbis, ybis, n_max, n = 0, mw, ntime = 0, nsum=0;
                Pen pen = new Pen(c, npen);
                Font fn = new Font("Verdana", 48);
                Brush br = new SolidBrush(Color.Black);
                String txt;
                Boolean b_result;

                x = 0;
                xvon = xX(x); n_max = this.ClientRectangle.Width;
                serialPort1.ReadExisting();
                serialPort1.Write(do_char);
                wait_ms(200);
                b_result = int.TryParse(serialPort1.ReadLine(), out mw);
                if (b_result)
                    yvon = yY(mw);
                else
                    yvon = yY(0);
                xbis = xvon + dn;
                while (xvon < n_max)
                {
                    b_result = int.TryParse(serialPort1.ReadLine(), out mw);
                    if (b_result)
                        ybis = yY(mw);
                    else
                        ybis = yY(0);
                    g.DrawLine(pen, xvon, yvon, xbis, ybis);
                    xvon = xbis; yvon = ybis;
                    xbis = xvon + dn;
                    //-----------------
                    if (get_zero_of_A0)
                    {
                        nsum += mw; n++;
                    }
                    //-----------------
                }
                if (get_zero_of_A0) hier_0 = nsum / n;
                //--------------------Time-------------
                serialPort1.Write("z");
                serialPort1.ReadExisting();
                wait_ms(100);
                serialPort1.ReadExisting();
                wait_ms(100);
                txt = "";
                for (i = 0; i < 3; i++)
                    txt += serialPort1.ReadLine();
                //g.DrawString(txt, fn, br, 50, 150);
                if (txt.Contains("begin") && txt.Contains("end"))
                {
                    String snum = txt.Remove(0, 5);
                    snum = snum.Remove(snum.IndexOf("end"));
                    b_result = int.TryParse(snum, out ntime);
                }
                if (b_result) g.DrawString("v = " + (5000000.0 / ntime).ToString("0.000") +
"cm/s", fn, br, 50, 50);
                //------------------------------------
                this.BeginInvoke(new EventHandler(closecom));
            }
            draw_A0 = get_zero_of_A0 = false;
        }
```

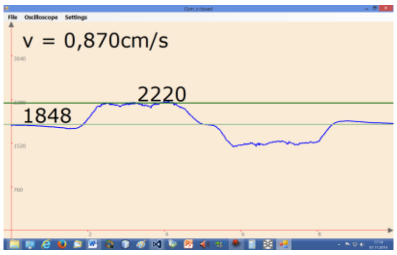
With less or more current we change the velocity of the coil, moving over the magnetic field.

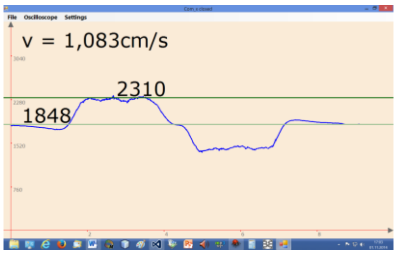


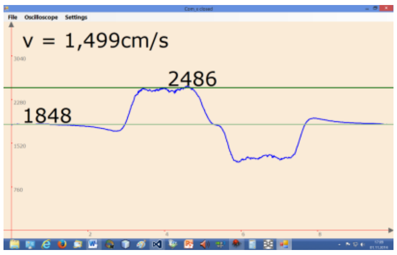


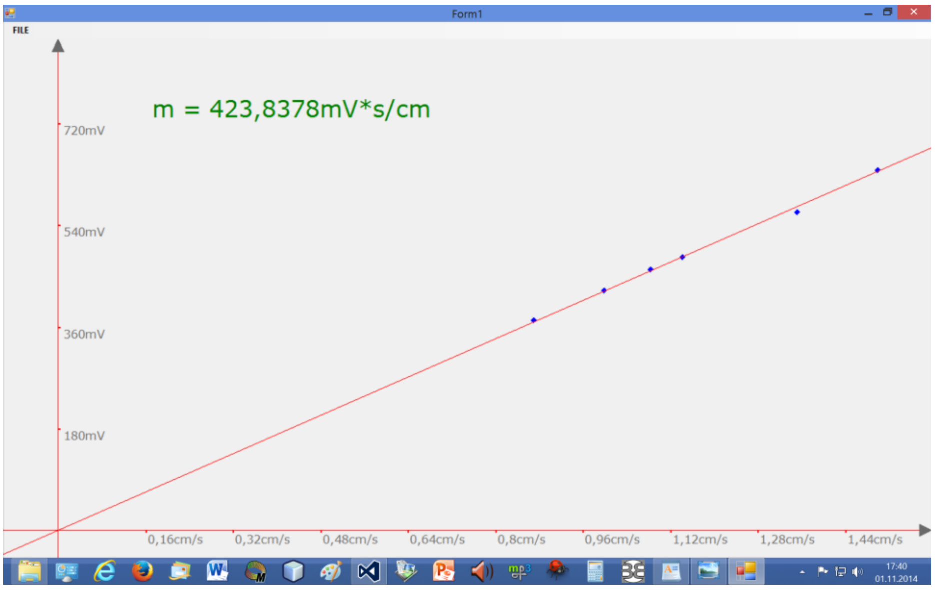
Some pictures of several measurements ...







... and the result was:



Thank you to the Arduino DUE team and their great work.

edgarmarx@t-online.de