Magnetic Field Meter

Introduction:

Magnetic field meters are devices which allow to measure magnetic flux density and magnetic field exposure [1]. These are often used to determine the magnetic anomalies or dipole moments of various materials and magnets.

This implementation makes use of an Arduino microcontroller as well as the ss49e Hall effect sensor to output a voltage proportional to the strength of the magnetic field being measured. The implementation's software allows to connect a computer through USB with the Arduino, which through serial communication sends the data. The software displays the data in real time as it is sent by the microcontroller.

Materials:

1. 1x Arduino Nano
2. 1x ss49e Hall Sensor

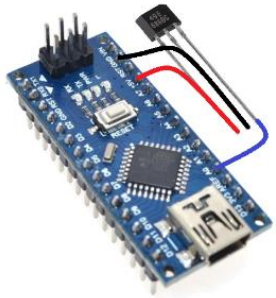

Illustration 1: Magnetic Field Meter

Component Explanation:

ss49e Sensor:

The ss49e is a hall effect sensor which outputs a voltage proportional to the external magnetic field. This sensor has a p-type semiconductor which has a current flowing through it. When the sensor is placed in proximity to a magnetic field, the field interacts with the semiconductor material deflecting the charge to one side of the material. As such, this creates a potential difference between the 2 sides of the semiconductor, resulting in a voltage proportional to the strength of the magnetic field [2].

The Hall effect voltage output can be calculated using the following formula:

$$Vh = Rh * \left(\frac{I}{t} * B\right) \qquad (1)$$

Where Vh is the Hall effect voltage output, Rh the Hall effect coefficient, I the current passing through the sensor in Amps, t the thickness of the sensor and B the magnetic flux density. The ss49e operate at 4.5-6V and provides a linear output of approximately 15mV/mT at room temperature.

Arduino:

An Arduino is a microcontroller board based on the ATmega328P. It has an unregulated power source in the VIN pin which supports 7-12V, and a regulated 5V power source [3]. For this implantation the Arduino is the device which reads the analog output of the ss49e sensor and sends it through serial communication to the computer through USB.

Final Device Diagram:

The device diagram is shown as follows:

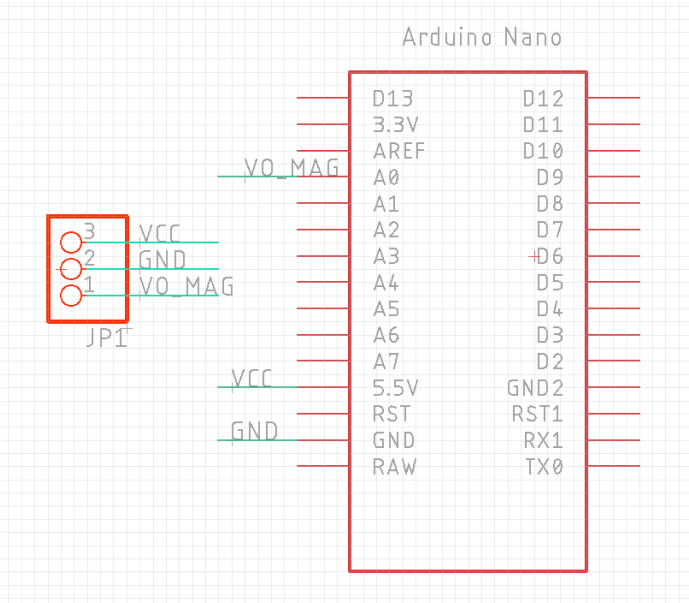

Illustration 2: Magnetic Field Meter Schematic

Implementation:

For the implementation, the following PCB is designed for fast and easy use and deployment of the device.

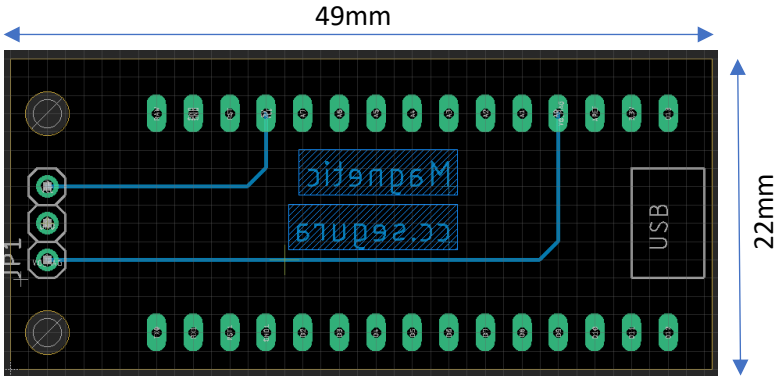

Illustration 3: Magnetic Field Meter PCB Design

Calibration:

Although no calibration is implemented, the methodology will be shown. It should be noted that for the device to be properly calibrated a magnet with a known magnetic field amplitude is needed.

The calibration methodology consists of a regression through which the magnet is measured at different distances to create a calibration equation. For point charges, the magnetic field strength follows Coulomb´s Law and as such is inversely proportional to D^2. For electric dipoles however, it is inversely proportional to D^3. Thus, data points should be measured to replicate an exponential behavior [4].

Once the calibration equation has been calculated, it can be implemented in the Arduino code. It should be noted that the magnetic field measurements should be conducted at a specified distance, which can be determined after the device's calibration.

Code:

Arduino:

The Arduino code consists a loop which reads the analog voltage in analog pin 0, as well as in analog pin 1. For this implementation, analog pin 1 should be grounded, so that the readings from the A1 pin are 0. The Arduino code can be observed in Annex 1.

Computer:

The software consists of a chart which shows the data obtained by the Arduino microcontroller in real time. If the A1 pin in the Arduino is grounded, the chart will only show the data relevant for the device, which is the data obtained by the A0 pin. The software's code can be observed in Annex 2.

The software implemented scans for COMs in the USB ports of the computer to try to connect. Once an Arduino COM is selected and the Connect button is clicked, it starts the serial communication with the Arduino microcontroller, which starts to send the data to the computer. The software features two text boxes which show the voltage readings from the A0 and A1 pins; these values are then mapped in the chart in real time. The software also features a Clear button which deletes the data in the chart; the chart, however, will continue to load data in real time as long as the Arduino sends it.

The following illustration shows the software implementation for this device.

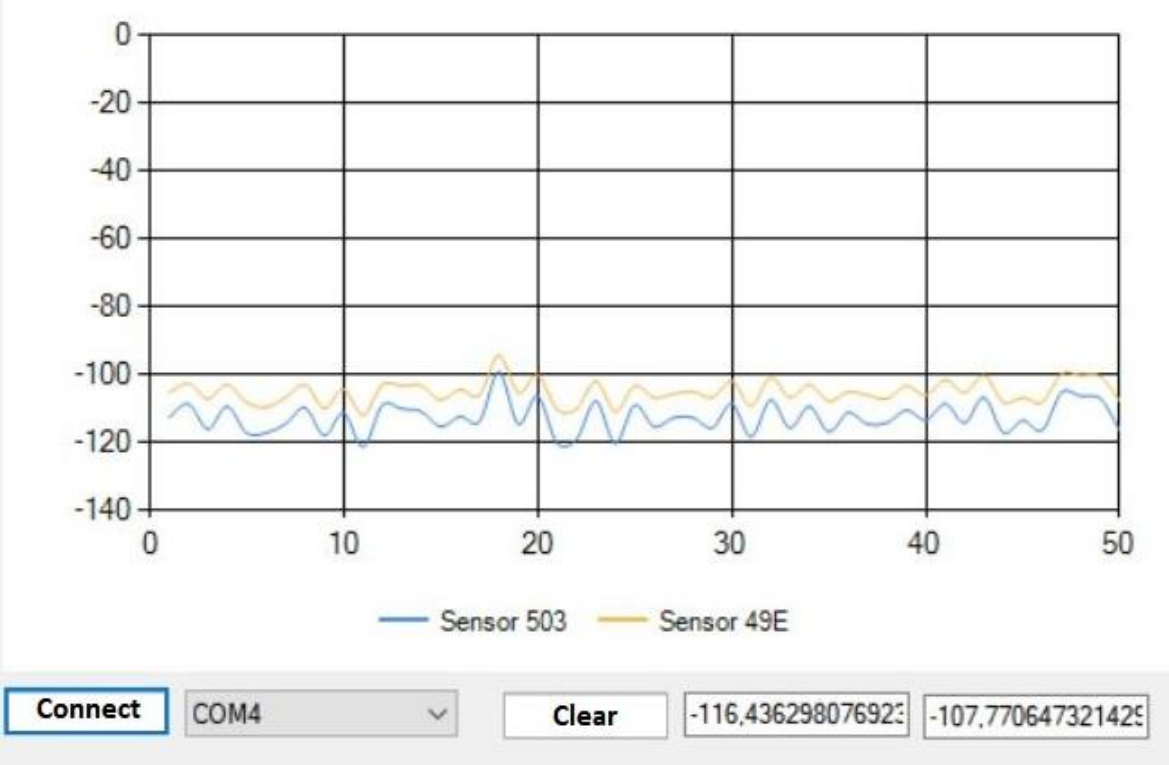

Illustration 4: Software Implementation

Recommendations:

Regarding the device's operation, the Arduino microcontroller can be affected by the magnetic field measured by the ss49e sensor. If the magnetic field is strong enough, the microcontroller may experience voltage fluctuations and as such may restart or shut down. If the implementation is to measure large magnetic fields, approximately 1T, it may be necessary to account for these fluctuations by moving the sensor further away from the Arduino Nano to diminish the magnetic field strength.

It should also be noted that both the Arduino as well as the software monitor the voltage in pins A0 and A1. Thus, if only one of the pins is to be used, the other should be grounded to reduce the voltage variations measured by the microcontroller. This will also allow to properly see the data in the software chart, as voltage variations from an unused pin can cause visual noise in the chart.

References:

1. *MAGNETIC FIELD HiTESTER FT3470-51*. HIOKI. (n.d.). Retrieved September 9, 2021, from https://www.hioki.com/global/products/field-measuring/magnetic-field/id_5815#:~:text=Hioki%20magnetic%20field%20meters%20are,research%20on%20magnetic%20field%20exposure.
2. *Hall Effect Sensor*. Basic Electronics Tutorials. (2018, February 9). Retrieved September 9, 2021, from https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html.
3. *¿Qué es Arduino?* Arduino. (2017, July 12). Retrieved September 9, 2021, from https://arduino.cl/que-es-arduino/.
4. *MAGNETIC FIELDS VARYING AS AN INVERSE CUBE*. Millersville University. (1970, September 9). Retrieved September 9, 2021, from https://www.millersville.edu/physics/experiments/023.

Annexes:

Arduino Code

```
void setup() {
  Serial.begin(9600);
}
int val = 0;
void loop() {
  if (Serial.available() > 0)
  {
    Serial.read();
    Serial.print("Val1:");
    val = analogRead(A0);
    Serial.print(val);
    Serial.print(":");
    val = analogRead(A1);
    Serial.print(val);
    Serial.println(":0");
  }
}
```

Software Code

```
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;
using System.IO.Ports;
using System.Threading;

namespace App_Hall
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            CheckForIllegalCrossThreadCalls = false;
            //this.Text = "" + (((180 * 5000 / 1024) - 2500) / 1.4 * 0.1);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
```

```
        {
            if (serialPort1.IsOpen)
                serialPort1.Close();

            serialPort1.PortName = comboBox1.SelectedItem.ToString();

            serialPort1.Open();
            timer1.Start();
        }
        catch (Exception ex)
        {
        }
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        var s = SerialPort.GetPortNames();
        foreach (String r in s)
        {
            comboBox1.Items.Add(r);
        }
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            serialPort1.Write("@");
        }
    }

    private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
    {
        try
        {
            SerialPort sp = (SerialPort)sender;
            Thread.Sleep(50);
            string indata = sp.ReadExisting();
            indata = indata.Replace('.', ',');
            var trama = indata.Split(':');
            if (trama.Length >= 3)
            {
                chart1.Series[0].Points.AddY(((double.Parse(trama[1]) * 5000 / 1024) - 2500) / 1.3 *
0.1);
                chart1.Series[1].Points.AddY(((double.Parse(trama[2]) * 5000 / 1024) - 2500) / 1.4 *
0.1);
                textBox1.Text = "" + ((double.Parse(trama[1]) * 5000 / 1024) - 2500) / 1.3 * 0.1;
                textBox2.Text = "" + ((double.Parse(trama[2]) * 5000 / 1024) - 2500) / 1.4 * 0.1;
```

```
            }
            if (chart1.Series[1].Points.Count > 50)
            {
                chart1.Series[0].Points.RemoveAt(0);
                chart1.Series[1].Points.RemoveAt(0);
            }
        }
        catch (Exception ex)
        {

        }
    }

    private void button1_Click_1(object sender, EventArgs e)
    {
        chart1.Series[0].Points.Clear();
        chart1.Series[1].Points.Clear();
    }
  }
}
```