Mech 368 Lab 3

Thomas Richmond, 52228848

21 October 2021

# Exercise 1

*Construction performed according to lab manual*

# Exercise 2

## Goals

* Implement an op-amp

## Setup



Diagram, engineering drawing

Description automatically generatedThe MCP6002 circuit diagram was obtained from the datasheet and is shown on the right.



To achieve given an input voltage of 5V, we get the following relation:

Let’s use a standard resistor value of . The circuit is created as shown:

## Data

The output from is measured using the oscilloscope. The trace is shown below.

A screenshot of a computer

Description automatically generated

The trace is very steady with a nominal voltage of 2,5V.

# Exercise 3

## Goals

* Understand the principle of a strain gauge using a mock-up

## Setup

## Data

The signals traces for and are very stable. At a point in time, we see the following output:

Chart

Description automatically generated with medium confidence

We see the signal is within one major division, corresponding to a difference in signals of

which is compliant with the requirements of this exercise.

# Exercise 4

## Goals

* Analyze a Wheatstone bridge

## Setup

The questions posed in the exercise are answered below:



Text

Description automatically generated with medium confidence

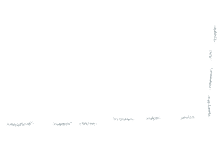


Graphical user interface, text, application, email

Description automatically generated

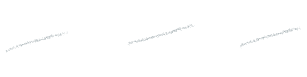


*Work shown on next page*

Diagram

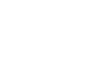
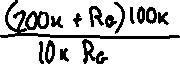
Description automatically generated











## Data

The signal amplification is calculated from the following trace, given the blue trace is for the strain gauge and the yellow trace is the amplifier output voltage .

Using the voltmeter function of the AD2, we find the following data:

|  |  |
| --- | --- |
| **Data** | **Value [mV]** |
|  | 3974 |
|  | 2502 |
|  | 2488 |
|  | 2495 |

Plugging these values into the equation above yields which is very close to our target value!

The no-load and maximum-load outputs are measured to be 2604mV and 3031mV, respectively. The difference in measurements is 427mV, which is commensurate with the values in the lab report.

# Exercise 5

## Goals

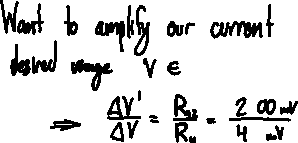
* Implement an offset amplifier to generate a reference voltage

## Setup



Diagram, schematic

Description automatically generated



## 



The circuit was tuned from the initial setup above such that the final nominal resistance of resistor series is .

The final circuit setup is as shown:

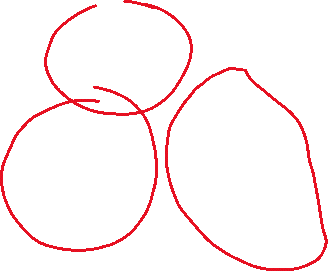
A close-up of a circuit board

Description automatically generated with medium confidence

With annotations:

A close-up of a circuit board

Description automatically generated with medium confidence



## Data

The no-load and maximum-load outputs are measured to be 2498mV and 460mV, respectively. This commensurate with the respective datapoints of 2500mV and 500mV as listed in the lab report.

# Exercise 6

## Goal

* Develop a program to visualize

## Setup

The code used was largely the same as from Lab 2 Exercise 6, with some minor modifications.

## Data

The output for the program is seen below. The graph contains some troughs, corresponding to probing of the load cell with my finger.

Table

Description automatically generated

The code for Exercise 6 is further developed in Exercise 8. Refer to Appendix A for the code for Exercise 8 (code contained for this exercise is embedded therein)

# Exercise 7

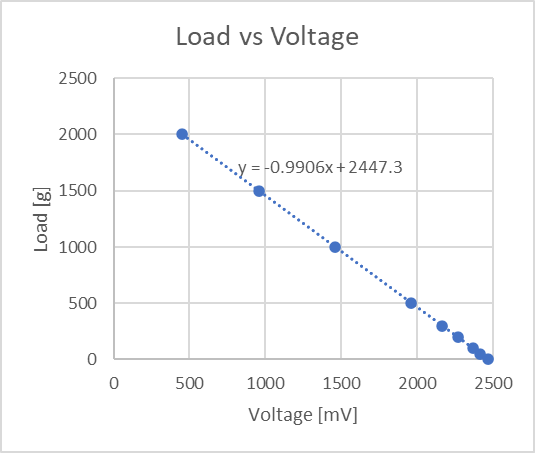
## Goal

* Fit a trendline to a load cell to convert voltage to load

## Setup

A trendline is developed by placing a variety of weights onto the load cell and recording the voltage output from the offset amplifier. This is performed using Excel.

## Data

The datapoints are plotted on a scatterplot and a trendline is fitted. A slope of is obtained.

|  |  |
| --- | --- |
| **Load [g]** | **Voltage [mV]** |
| 0 | 2473 |
| 50 | 2418 |
| 100 | 2369 |
| 200 | 2270 |
| 300 | 2167 |
| 500 | 1965 |
| 1000 | 1461 |
| 1500 | 956 |
| 2000 | 452 |

# Exercise 8

## Goal

* Develop a UI to display load from a load cell
* Implement a stability detection algorithm

## Setup

The code from Exercise 6 is modified slightly to incorporate the following features:

* The average voltage is multiplied by the slope above to generate a load reading
* Tare functionality is added in two ways:
  + A tare button is added to the UI, allowing for the user to tare the scale at any time
  + The tare is automatically set the first time a signal is read by the program
* A stability detection algorithm is implemented; the algorithm generates a standard deviation over the last five hundred milliseconds, and if it is below 10 (empirically determined) then the reading is considered stable

## Data

The program functions very well. When the load cell is loaded to 500g the load is reported to be between 498 and 502 grams, falling within 1% error.

Table

Description automatically generated

A picture containing indoor, cluttered, equipment

Description automatically generated

# Appendix A: Exercise 8 Code

|  |
| --- |
| using System;  using System.IO;  using System.IO.Ports;  using System.Windows.Forms;  using System.Collections.Generic;  using System.Collections.Concurrent;  using System.Windows.Forms.DataVisualization.Charting;  namespace Mech368Lab2E6  {  public partial class Mech368L3E6 : Form  {  enum AD2Token  {  PACKET\_HEADER = 0xFF,  STREAM\_REQUEST\_ACCELEROMETER = 'A',  STREAM\_REQUEST\_THERMISTOR = 'c',  LED5\_ON = '5',  LED5\_OFF = '%'  }  private ConcurrentQueue<int> streamBuffer;  private SerialPort port;  private StreamWriter streamWriter;  private Series loadvoltages;  private Series loads;  private double load = 0;  private double tare = double.NaN;  public Mech368L3E6()  {  InitializeComponent();  InitializeListeners();  this.streamBuffer = new ConcurrentQueue<int>();  this.port = this.BindSerial();  this.port.Write(((char)AD2Token.STREAM\_REQUEST\_THERMISTOR).ToString());    this.loadvoltages = new Series("Load Voltages");  this.loadvoltages.ChartType = SeriesChartType.FastLine;  this.loadvoltages.YValueType = ChartValueType.Double;  //plot.Series.Add(this.loadvoltages);  this.loads = new Series("Load");  this.loads.ChartType = SeriesChartType.FastLine;  this.loads.YValueType = ChartValueType.Double;  plot.Series.Add(this.loads);    //this.tareBtn.Visible = false;  //this.stabilityLabel.Visible = false;  //this.loadTB.Visible = false;  //this.label3.Visible = false;  //this.label4.Visible = false;  }  private void InitializeListeners()  {  this.filepathTB.Click += this.ConfigureOutputStream;  this.savetofileCB.CheckedChanged += this.ToggleFilewriter;  this.processLoop.Tick += this.ProcessDatastream;  this.processLoop.Tick += this.DetectStability;  this.tareBtn.Click += this.SetTare;  }    private SerialPort BindSerial()  {  SerialPort p = new SerialPort()  {  PortName = "COM3",  BaudRate = 9600,  Parity = Parity.None,  DataBits = 8,  StopBits = StopBits.One,  Handshake = Handshake.None  };  p.DataReceived += this.CaptureDatastream;  p.Open();    return p;  }  private void ConfigureOutputStream(object sender, EventArgs e)  {  SaveFileDialog sfd = new SaveFileDialog();  if (sfd.ShowDialog() == DialogResult.OK)  filepathTB.Text = sfd.FileName;  }  private void ToggleFilewriter(object sender, EventArgs e)  {  if (this.streamWriter == null)  if (this.filepathTB.Text.Length > 0)  this.streamWriter = new StreamWriter(this.filepathTB.Text);  else  MessageBox.Show("Invalid filename!");  else  {  this.streamWriter.Close();  this.streamWriter = null;  }  }  private void OnFormClosing(object sender, EventArgs e)  {  if (this.port != null && this.port.IsOpen)  this.port.Close();  if (this.streamWriter != null)  this.streamWriter.Close();  }  private void CaptureDatastream(object sender, EventArgs e)  {  int next; // Byte buffer  try  {  while ((next = this.port.ReadByte()) != -1)  this.streamBuffer.Enqueue(next); // ConcurrentQueue buffer implementation  }  catch { /\*Ignored!\*/ }  }  Queue<int> circularBuffer = new Queue<int>();  private void ProcessDatastream(object sender, EventArgs e)  {  void EnqueueCircular(Queue<int> queue, int next, int max)  {  queue.Enqueue(next);  if (queue.Count > max)  queue.Dequeue();  }  int AverageQueue(Queue<int> queue, int offset = 0)  {  float avg = 0;  foreach (int val in queue)  avg += val - offset;  avg /= (float)queue.Count;  return (int) Math.Round(avg);  }  int signal = -1;  while (this.streamBuffer.TryDequeue(out int next))  {  if (next == (int)AD2Token.PACKET\_HEADER)  {  if (signal == -1)  {  signal = 0;  continue;  }  double discretized = signal / ((double)(2 << 9));  int nextVoltage = (int) (discretized \* 3600);  EnqueueCircular(this.circularBuffer, nextVoltage, 50);  int avgVoltage = AverageQueue(this.circularBuffer);  voltageTB.Text = nextVoltage.ToString();  voltageAvgTB.Text = avgVoltage.ToString();// avgVoltage.ToString();  this.loadvoltages.Points.AddY(nextVoltage);  double nextLoad = avgVoltage \* -0.9906;  if (double.IsNaN(this.tare))  this.tare = nextLoad;  nextLoad -= tare;  loadTB.Text = nextLoad.ToString();  this.loads.Points.AddY(load);  if (this.streamWriter != null)  this.streamWriter.WriteLine(  $"{DateTime.Now.ToString("hh.mm.ss.ffffff")}, "  + $"{nextVoltage}, "  + $"{avgVoltage}, "  + nextLoad.ToString("0.##"));    this.load = nextLoad;  signal = 0;  continue;  }  else if (signal == -1) // Haven't captured a packet header  continue;  else // Assemble bytes  signal = (signal << 5) + next;  }  }  private void DetectStability(object sender, EventArgs e)  {  double SumSquaresQueue(Queue<int> queue)  {  float summed = 0;  foreach (int val in queue)  summed += val \* val;  return summed;  }  double AverageQueue(Queue<int> queue, int offset = 0)  {  float avg = 0;  foreach (int val in queue)  avg += val - offset;  avg /= (float)queue.Count;  return avg;  }  double sum = SumSquaresQueue(this.circularBuffer);  double mean = AverageQueue(this.circularBuffer);  double stdev = Math.Sqrt(sum / ((double)this.circularBuffer.Count) - mean \* mean);  //this.filepathTB.Text = stdev.ToString();  this.stabilityLabel.Text = stdev < 10 ? "STABLE" : "UNSTABLE";  }  private void SetTare(object sender, EventArgs e)  {  this.tare = this.load + this.tare;  }  }  } |