

<https://www.halvorsen.blog>

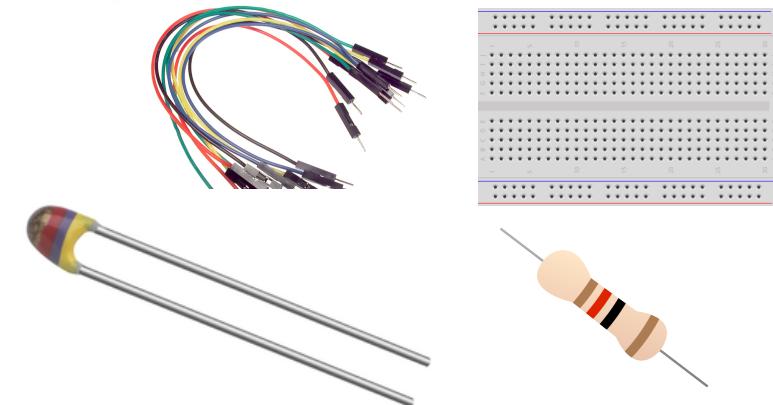


Thermistor

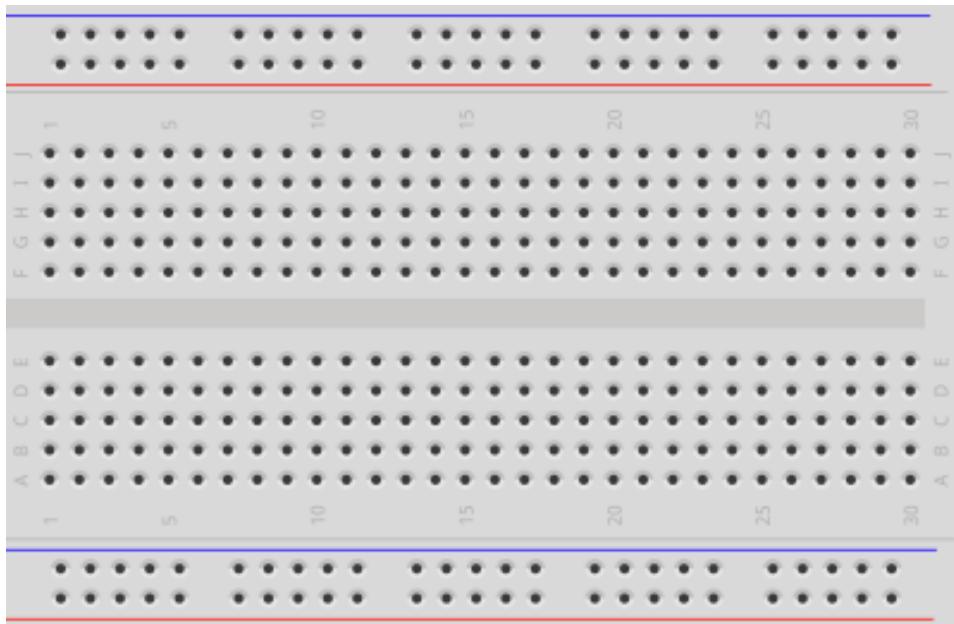
Hans-Petter Halvorsen

Hardware

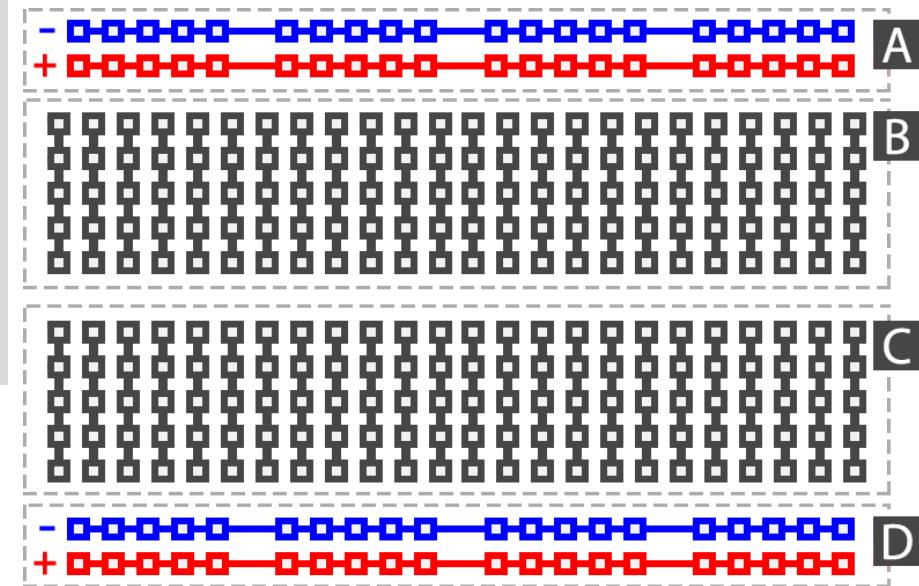
- DAQ Device (e.g., USB-6008)
- Breadboard
- Thermistor 10K (Temperature Sensor)
- Resistor, $R = 10k\Omega$
- Wires (Jumper Wires)



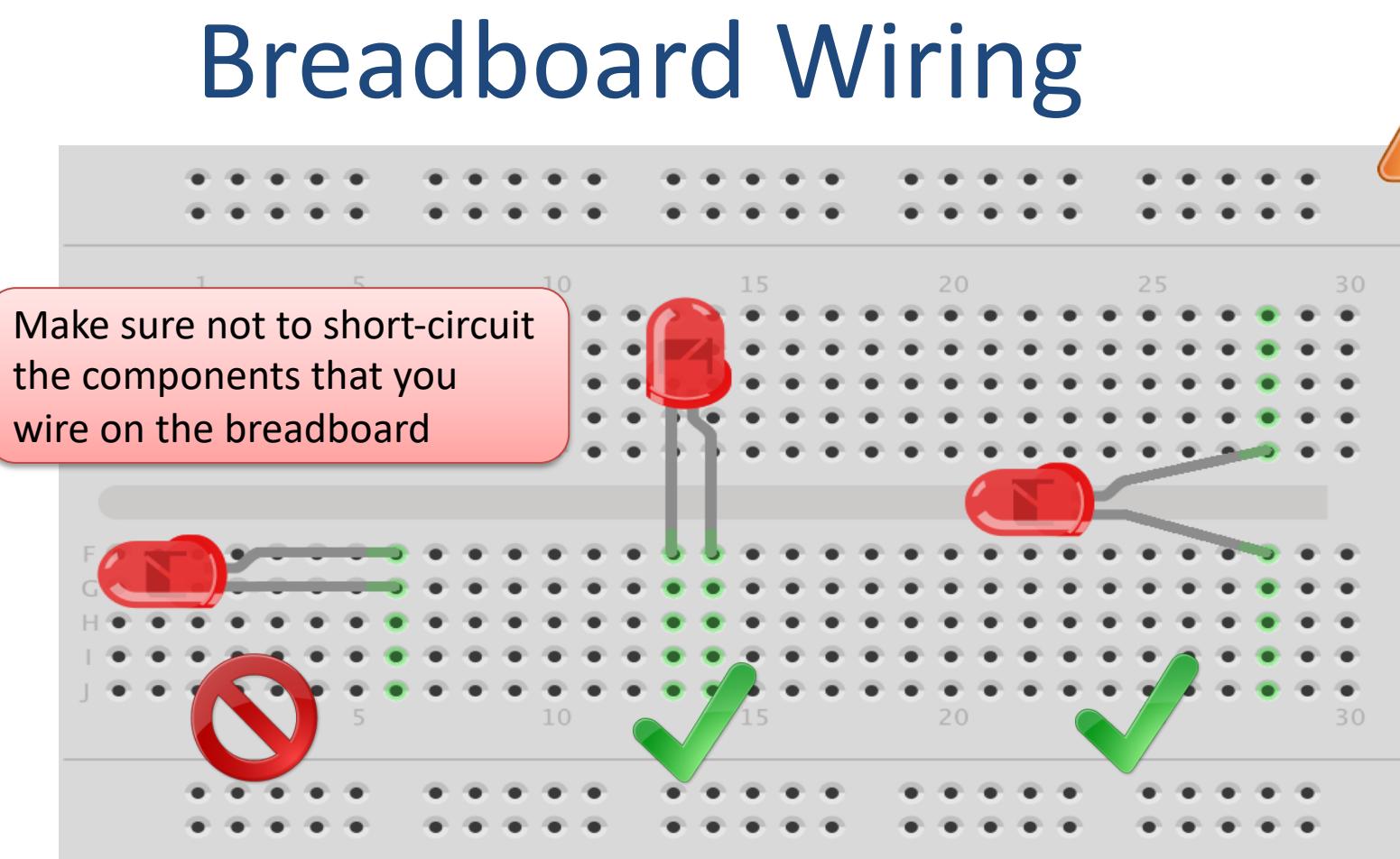
Breadboard



A breadboard is used to wire electric components together



Breadboard Wiring



The Breadboard is used to connect components and electrical circuits

fritzing

<https://www.halvorsen.blog>



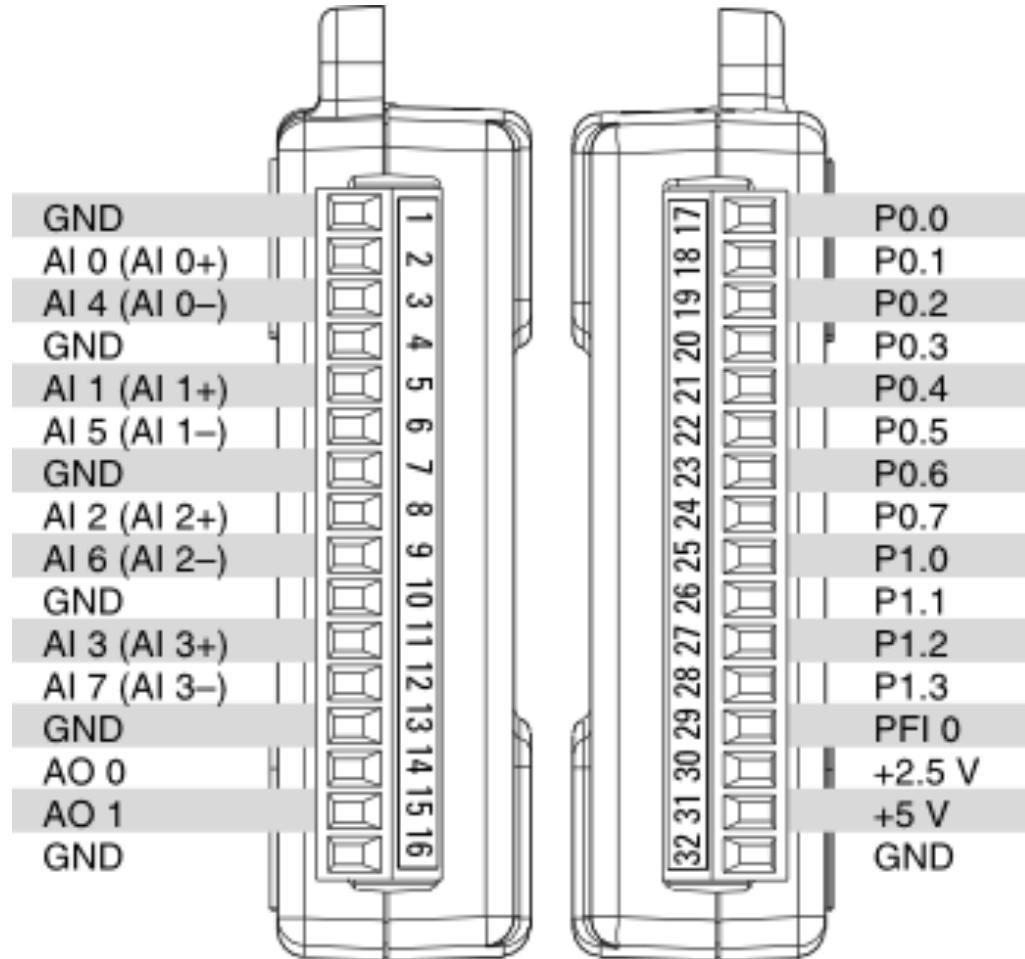
USB-6008

Hans-Petter Halvorsen

USB-6008



I/O Pins



<https://www.halvorsen.blog>



DAQmx

Hans-Petter Halvorsen

Measurement & Automation Explorer (MAX)

The screenshot shows the NI Measurement & Automation Explorer (MAX) interface. The left pane displays a tree view of system components under "My System". The "Devices and Interfaces" section lists several devices, including "NI USB-6008 "Dev1"" which is currently selected. Other listed devices include "Integrated Webcam "cam0"" and "Microsoft® LifeCam Studio(TM)". The right pane contains a "Settings" dialog box for the selected device. The "Settings" tab shows the following information:

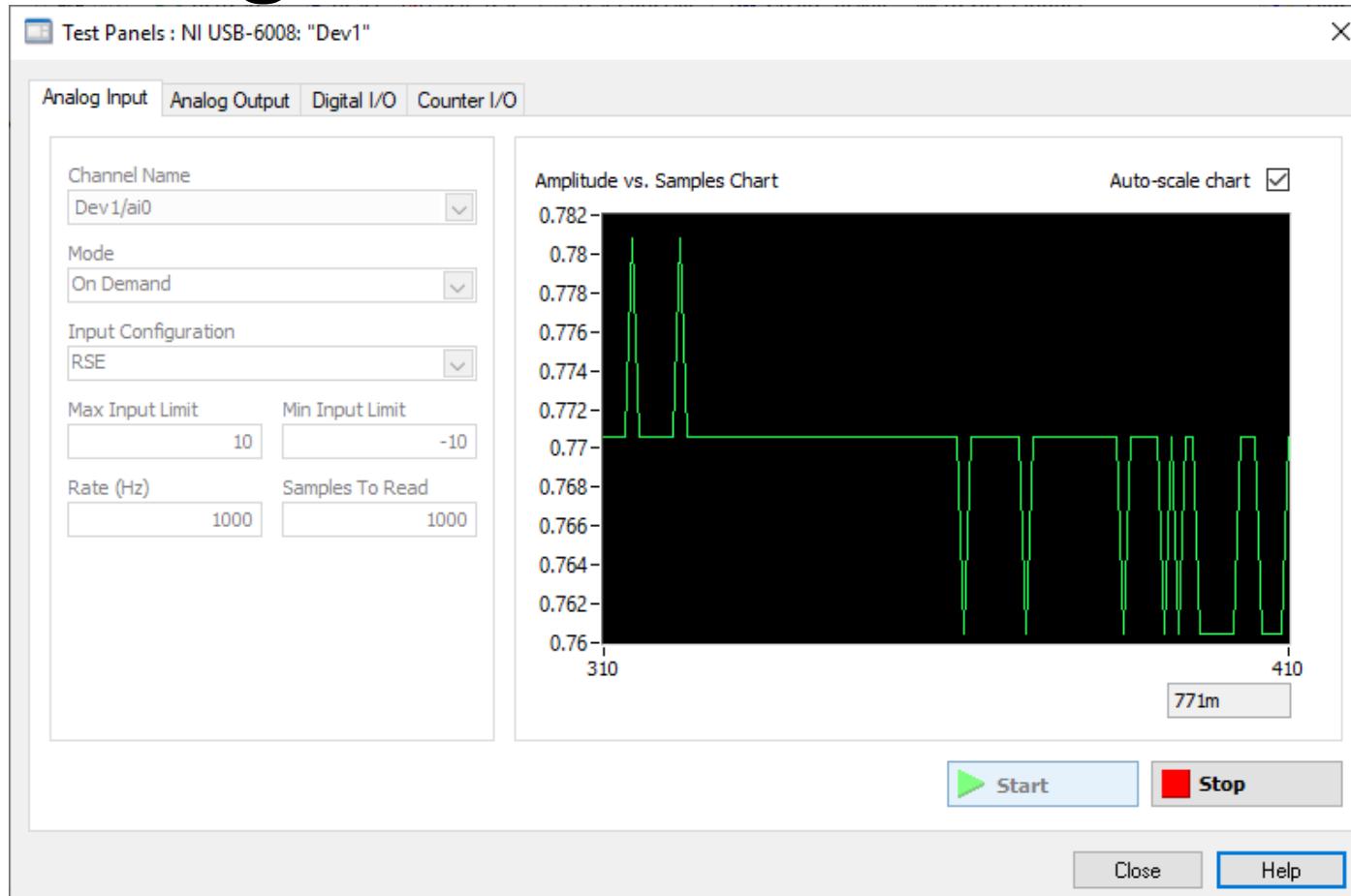
| | |
|---------------|----------------------|
| Name | Dev1 |
| Vendor | National Instruments |
| Model | NI USB-6008 |
| Serial Number | 0165408B |
| Status | Present |

Below the settings, there is an "External Calibration" section with the following data:

| | |
|------------------------------|------------------|
| Calibration Date | 2011-10-03 00:00 |
| Recommended Next Calibration | 2012-10-03 00:00 |

The top bar includes standard menu options like File, Edit, View, Tools, Help, and buttons for Save, Refresh, Reset, Self-Test, Test Panels..., Create Task..., and Hide Help. A message box in the center says "The self test completed successfully." The right sidebar provides links for "NI-DAQmx Device Basics", "Run the NI-DAQmx Test Panels", "Remove the device", and "View or change device configuration".

Using the Test Panel in MAX



<https://www.halvorsen.blog>



Thermistor 10k Ω

Hans-Petter Halvorsen



Thermistor

A thermistor is an electronic component that changes resistance to temperature - so-called Resistance Temperature Detectors (RTD). It is often used as a temperature sensor.

Our Thermistor is a so-called NTC (Negative Temperature Coefficient). In an NTC Thermistor, resistance decreases as the temperature rises.

There is a non-linear relationship between resistance and excitement. To find the temperature we can use the following equation (Steinhart-Hart equation):

$$\frac{1}{T} = A + B \ln(R) + C(\ln(R))^3$$

where A, B, C are constants given below

$$A = 0.001129148, B = 0.000234125 \text{ and } C = 8.76741E - 08$$

[Wikipedia]

Steinhart-Hart equation

To find the Temperature we can use Steinhart-Hart equation:

$$\frac{1}{T_K} = A + B \ln(R) + C(\ln(R))^3$$

This gives:

$$T_K = \frac{1}{A + B \ln(R) + C(\ln(R))^3}$$

$$\begin{aligned} A &= 0.001129148, \\ B &= 0.000234125 \\ C &= 0.0000000876741 \end{aligned}$$

Where the Temperature T_K is in Kelvin

A, B and C are constants

The Temperature in degrees Celsius will then be:

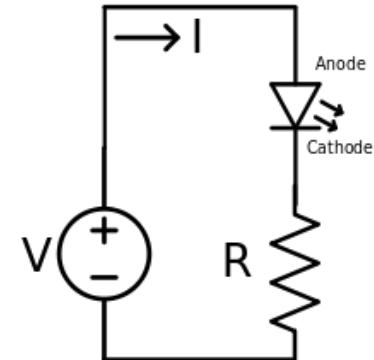
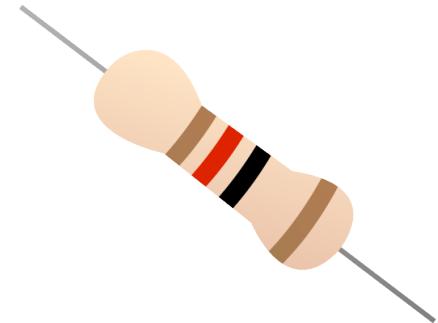
$$T_C = T_K - 273.15$$

Resistors

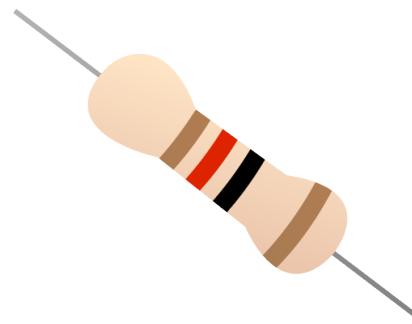
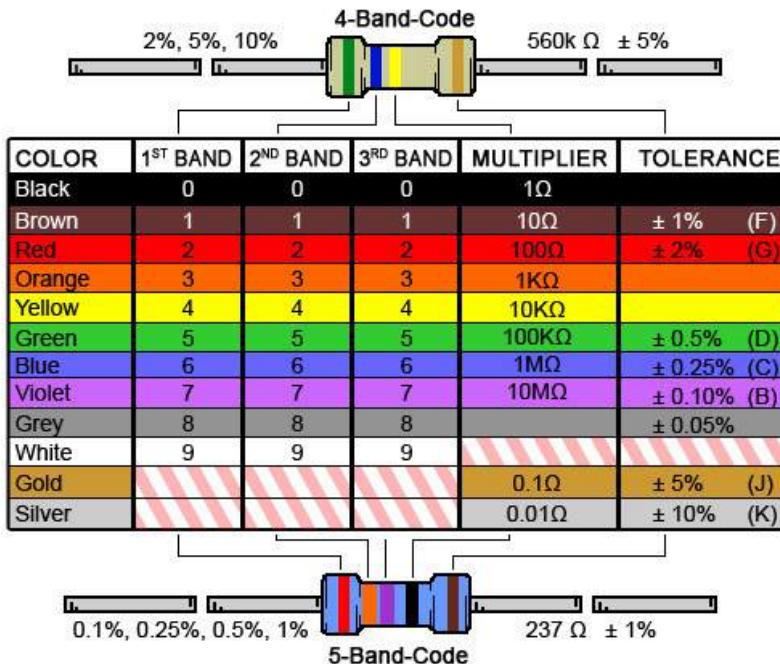
Resistance is measured in Ohm (Ω)

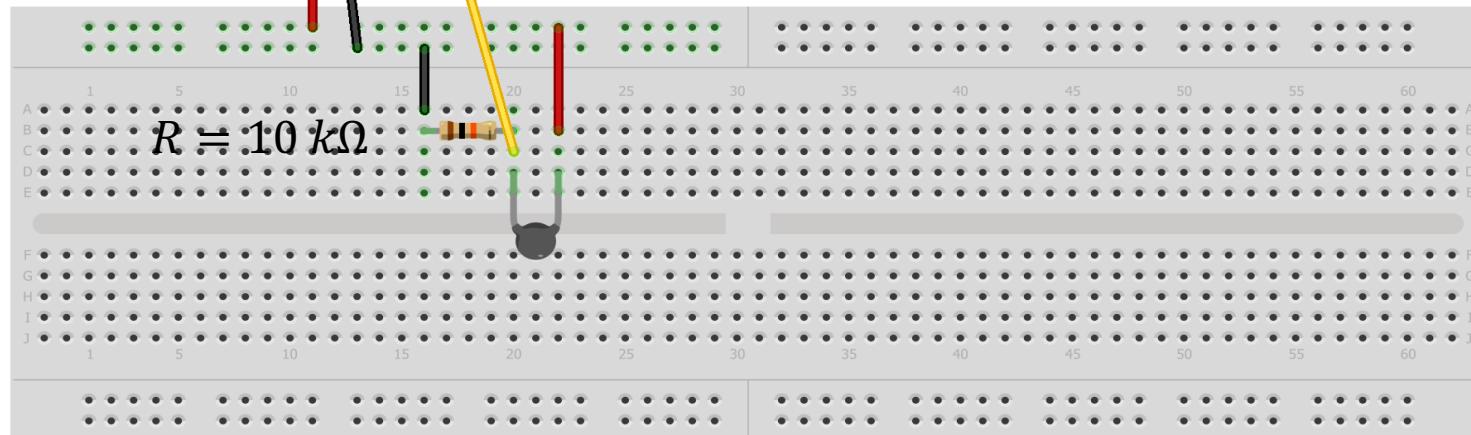
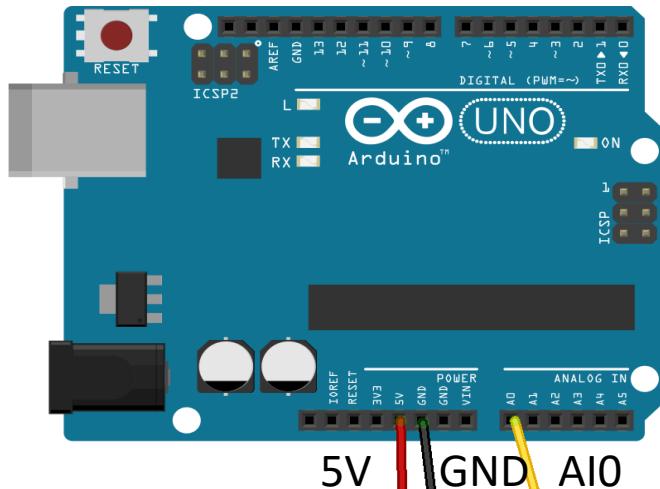
Resistors comes in many sizes, e.g., 220Ω ,
 270Ω , 330Ω , $1k\Omega$ $10k\Omega$, ...

The resistance can be found using Ohms Law
 $U = RI$



Resistor Color Codes

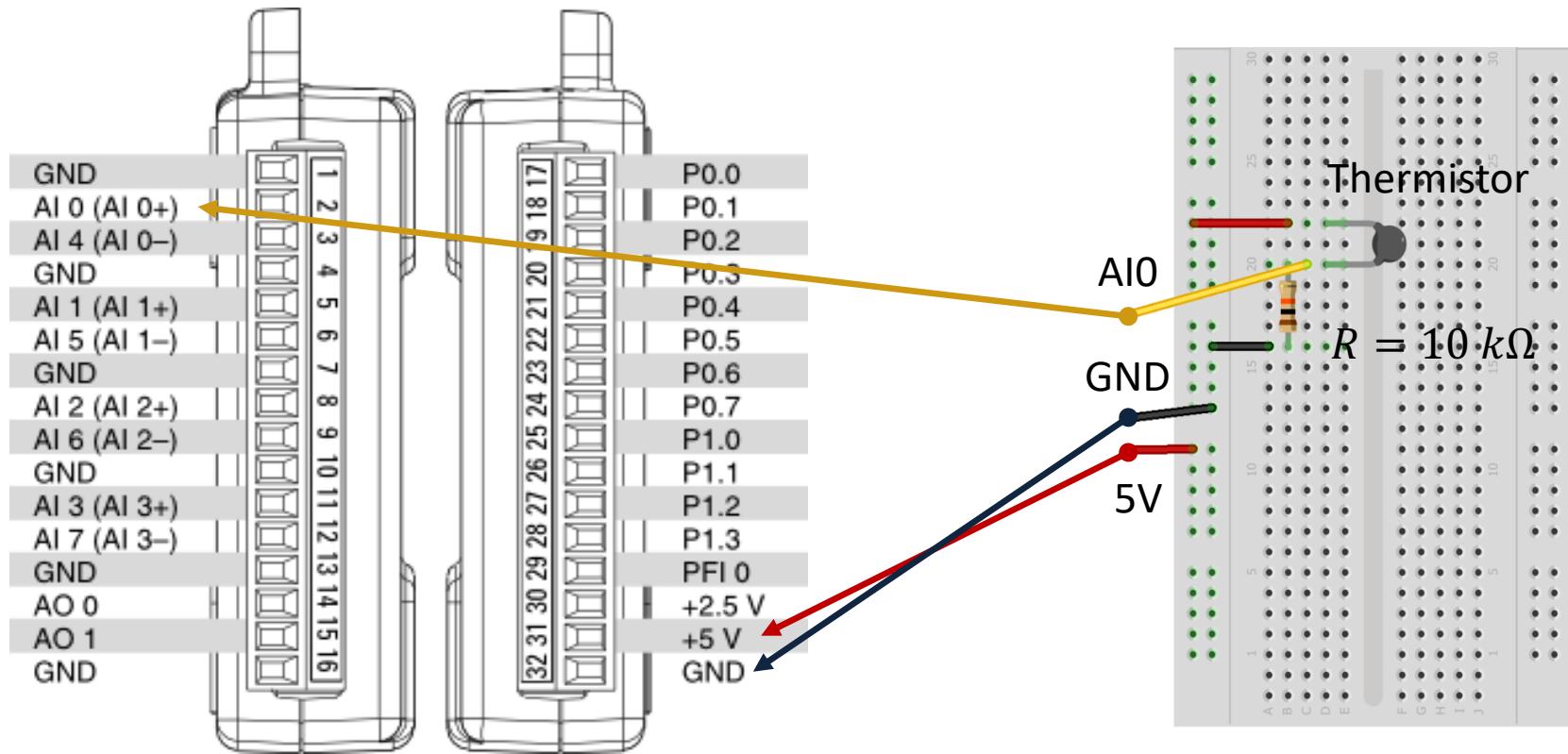




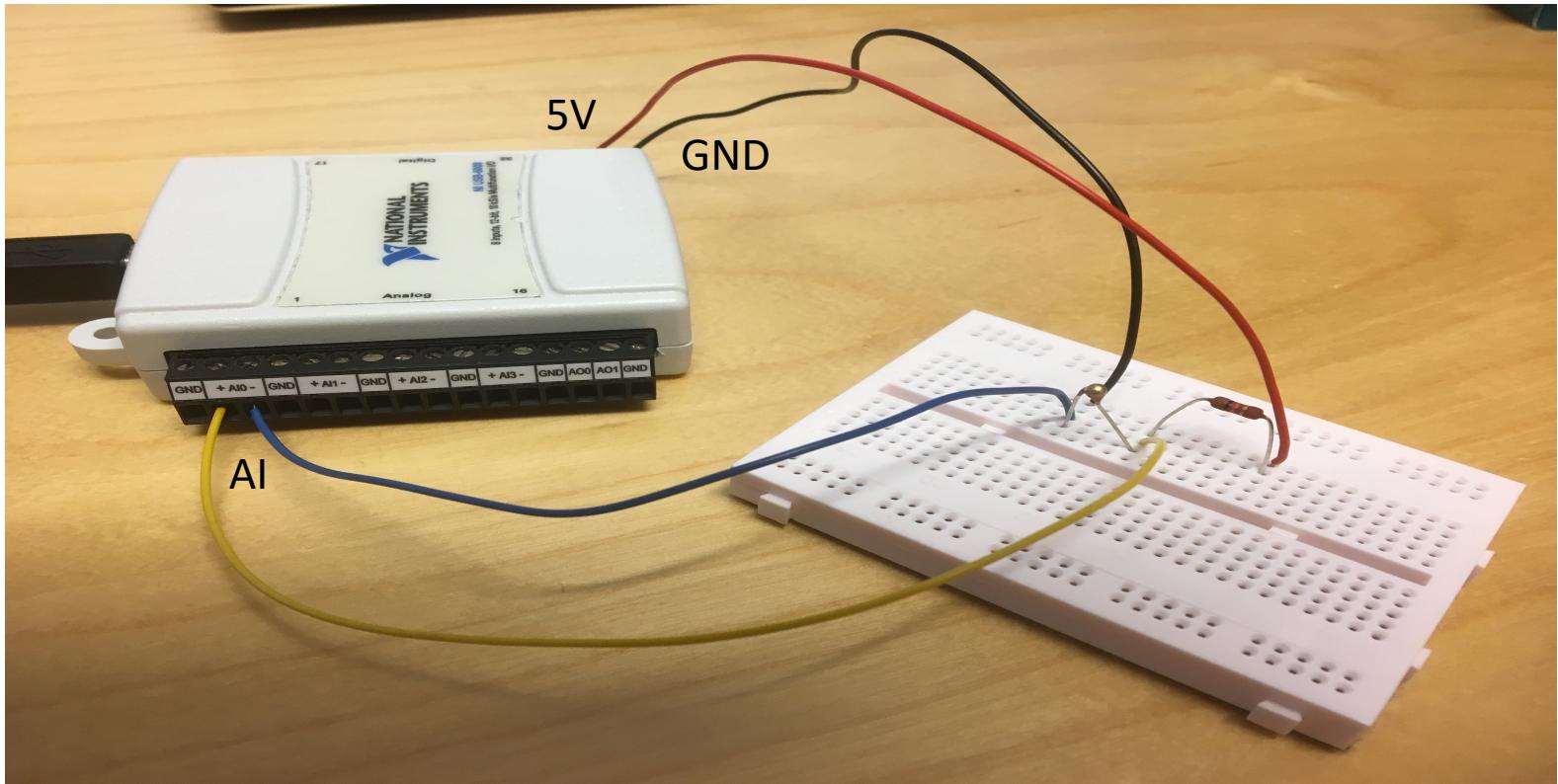
Wiring Example

Here you see a wiring examples using Arduino. You make the same wiring using a DAQ device like USB-6008 or similar.

Wiring Example



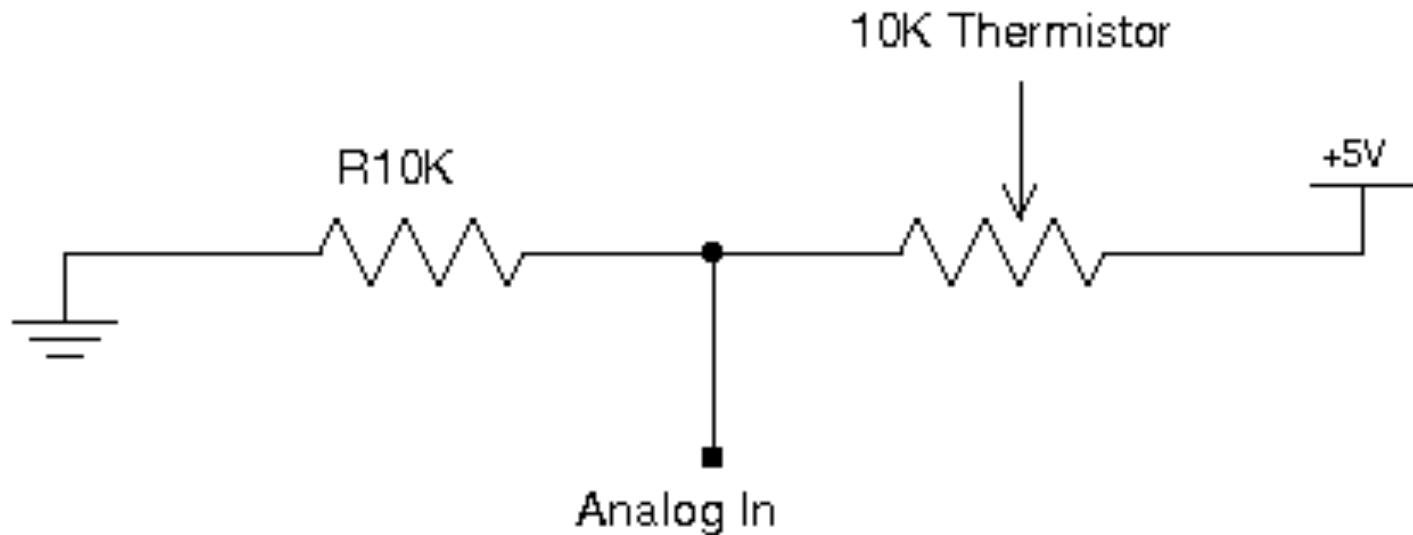
USB-6008 Wiring Example



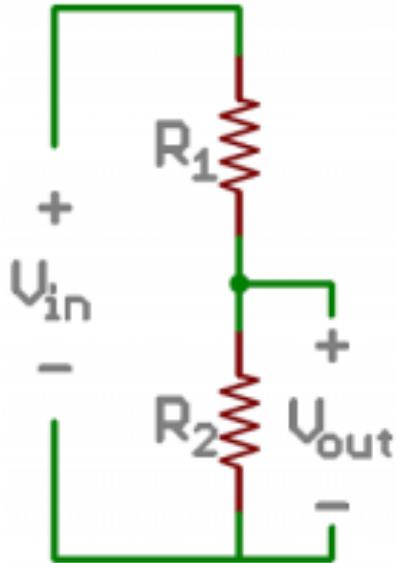
We connect the sensor to LabVIEW using a USB DAQ Device from National Instruments, e.g., USB-6001, USB-6008 or similar. A breadboard has been used for the wiring.

Wiring

The wiring is called a “Voltage divider”:



General Voltage Divider



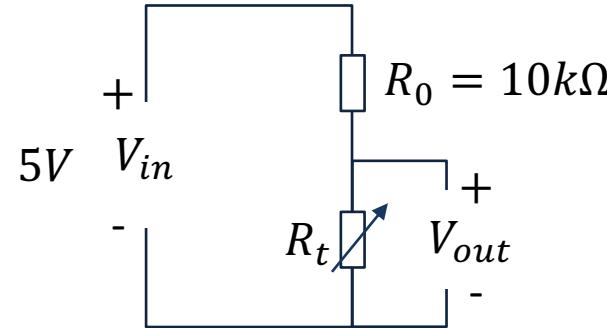
$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$

Voltage Divider for our system

Voltage Divider Equation:

$$V_{out} = V_{in} \frac{R_t}{R_0 + R_t}$$

We want to find R_t : $R_t = \frac{V_{out}R_0}{V_{in} - V_{out}}$



R_t - 10k Thermistor. This varies with temperature. From Datasheet we know that $R_t = 10k\Omega @ 25^\circ C$

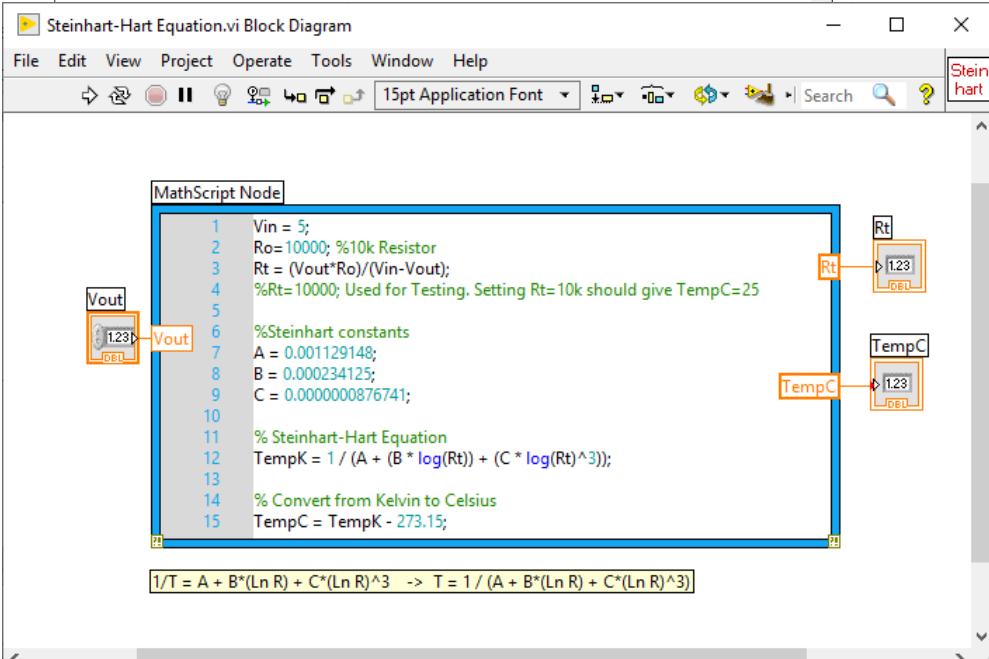
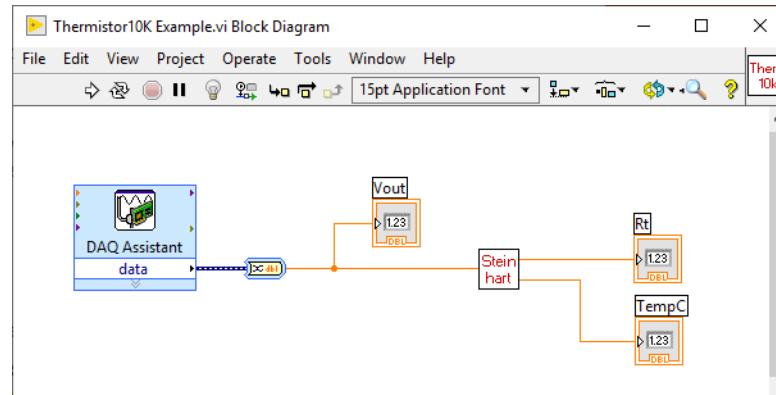
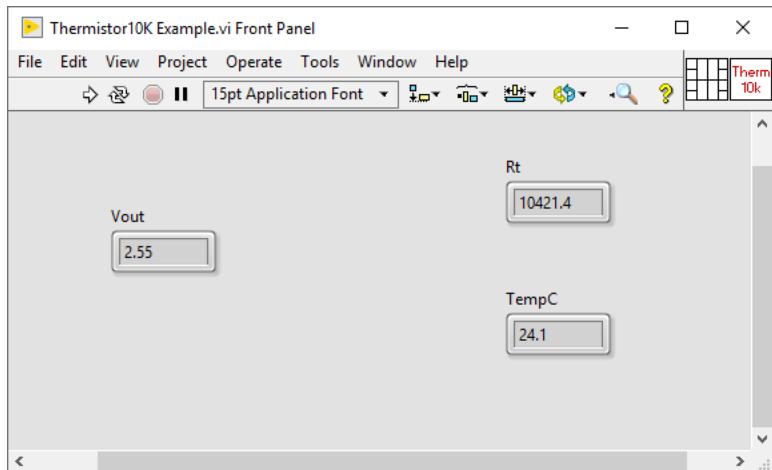
Steps:

1. We wire the circuit on the Breadboard and connect it to the DAQ device
2. We measure V_{out} using the DAQ device
3. We calculate R_t using the Voltage Divider equation
4. Finally, we use Steinhart-Hart equation for finding the Temperature

Code

1. Get V_{out} from the DAQ device
2. Calculate $R_t = \frac{V_{out}R_0}{V_{in}-V_{out}}$
3. Calculate $T_K = \frac{1}{A+B \ln(R_t)+C(\ln(R_t))^3}$
4. Calculate $T_C = T_K - 273.15$
5. Present T_C in the User Interface

LabVIEW Example



Arduino Example

```
const int temperaturePin = 0;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    int temperature = getTemp();
    Serial.print("Temperature Value: ");
    Serial.print(temperature);
    Serial.println("*C");
    delay(1000);
}

double getTemp()
{
    // Inputs ADC Value from Thermistor and outputs Temperature in Celsius

    int RawADC = analogRead(temperaturePin);
    long Resistance;
    double Temp;

    // Assuming a 10k Thermistor. Calculation is actually: Resistance = (1024/ADC)
    Resistance=((10240000/RawADC) - 10000);

    // Utilizes the Steinhart-Hart Thermistor Equation:
    // Temperature in Kelvin = 1 / {A + B[ln(R)] + C[ln(R)]^3}
    // where A = 0.001129148, B = 0.000234125 and C = 8.76741E-08

    Temp = log(Resistance);
    Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.000000876741 * Temp * Temp * Temp));
    Temp = Temp - 273.15; // Convert Kelvin to Celsius
    return Temp; // Return the Temperature
}
```

Celsius to Fahrenheit Conversion

In Norway we typically use Celsius as temperature unit, while in US they use Fahrenheit.

Conversion between these are as follows:

$$T_F = \frac{9}{5} T_C + 32$$

<https://www.halvorsen.blog>



Visual Studio

Hans-Petter Halvorsen

NI-DAQmx Driver

Installing

Select Agree Review Finish

Additional items you may wish to install:

| |
|---|
| <input checked="" type="checkbox"/> NI-DAQmx Runtime with Configuration Support Run-time components required to deploy applications using NI devices and support for configuring NI hardware via the Measu |
| <input checked="" type="checkbox"/> NI-DAQmx Support for .NET Framework 4.0 Languages Provides .NET interface for DAQ devices and adds NI-DAQmx su |
| <input checked="" type="checkbox"/> NI-DAQmx Support for .NET Framework 4.5 Languages Provides .NET interface for DAQ devices and adds NI-DAQmx su |
| <input checked="" type="checkbox"/> NI-DAQmx Support for C Provides files to create NI-DAQmx applications using ANSI C co |
| <input checked="" type="checkbox"/> NI-DAQmx Support for LabVIEW 2019 (32-bit) Provides NI-DAQmx support for LabVIEW 2019 (32-bit) |
| <input checked="" type="checkbox"/> NI-DAQmx Support for LabVIEW Real-Time and LabWindows/C Files used to create NI-DAQmx applications with LabVIEW Real-Real-Time Module. |
| <input type="checkbox"/> NI Linux RT PXI System Image |
| <input checked="" type="checkbox"/> NI-DAQmx cDAQ Firmware Provides firmware for Ethernet CompactDAQ Chassis, FieldDAQ, and NI Linux Real-Time CompactDAQ Controllers |
| <input type="checkbox"/> NI-DAQmx Support for Visual Studio 2015 DAQmx integration support for Microsoft Visual Studio 2015 |
| <input type="checkbox"/> NI-DAQmx Support for Visual Studio 2017 DAQmx integration support for Microsoft Visual Studio 2017 |
| <input checked="" type="checkbox"/> NI-DAQmx Support for Visual Studio 2019 DAQmx integration support for Microsoft Visual Studio 2019 |

Select All Deselect All

Next

Select Agree Review Finish

Additional items you may wish to install:

- NI-DAQmx Support for LabVIEW Real-Time and LabWindows/CVI Real-Time
Files used to create NI-DAQmx applications with LabVIEW Real-Time or with the LabWindows/CVI Real-Time Module.
- NI Linux RT PXI System Image
This software package includes the system image necessary for formatting and provisioning a supported PXI controller to run NI Linux Real-Time. Updated system images are required to install the latest drivers to the controller.
- NI-DAQmx cDAQ Firmware
Provides firmware for Ethernet CompactDAQ Chassis, FieldDAQ, and NI Linux Real-Time CompactDAQ Controllers
- NI-DAQmx Support for Visual Studio 2015
DAQmx integration support for Microsoft Visual Studio 2015
- NI-DAQmx Support for Visual Studio 2017
DAQmx integration support for Microsoft Visual Studio 2017
- NI-DAQmx Support for Visual Studio 2019
DAQmx integration support for Microsoft Visual Studio 2019

Select All Deselect All

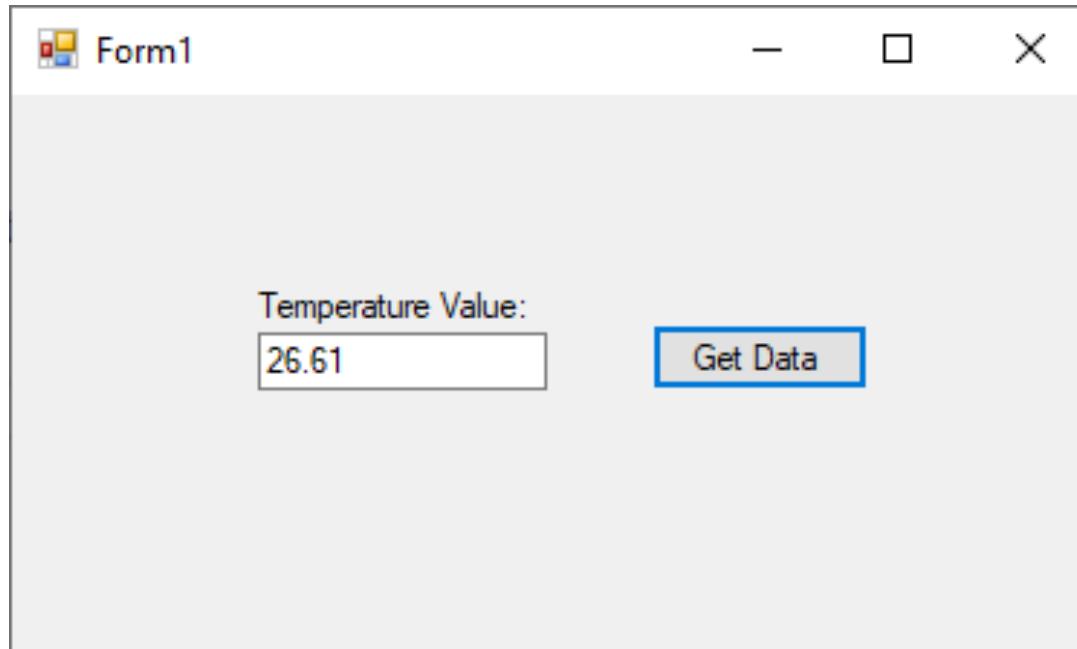
NI-DAQmx Examples

A screenshot of a Windows operating system interface. On the left, there is a vertical taskbar with icons for File Explorer, Task View, Start, and other system functions. A search bar at the bottom left contains the text "DAQ". The main area shows a search results pane on the left with "Best match" results: "NI-DAQmx Examples" (App) highlighted in blue, "NI-DAQmx Documentation", and "DAQ - See web results". The right side shows a detailed view of the "NI-DAQmx Examples" folder. The folder icon is a yellow folder with a blue ribbon. The title bar says "NI-DAQmx Examples App". The file list shows the following structure and files:

| Name | Date modified | Type | Size |
|-----------------|------------------|-------------|------|
| Analog In | 2019-06-11 09:11 | File folder | |
| Analog Out | 2019-06-11 09:11 | File folder | |
| Control | 2019-06-11 09:11 | File folder | |
| Counter | 2019-06-11 09:11 | File folder | |
| Digital | 2019-06-11 09:11 | File folder | |
| Synchronization | 2019-06-11 09:11 | File folder | |

The file list also includes a "File" menu with options like Open, Home, Share, View, and a context menu with items such as Quick access, Creative Cloud Files, OneDrive - Personal, OneDrive - USN, This PC, 3D Objects, Desktop, Documents, Downloads, Music, Pictures, Videos, OS (C:), ExtraHDD (E:), ExtraHDD (E:), and Network.

We will make the following Application:



Visual Studio 2019

Open recent

Today

WeatherSystem.sln 2019-12-18 15:46
C:\Users\hansha\OneDrive\Programming\Weather System\WeatherSystem

Tmp36Demo.sln 2019-12-18 11:48
C:\Temp\Development\Tmp36Demo

Tmp36App.sln 2019-12-18 11:05
C:\...\Sensors and Actuators Examples\TMP36\Code\Tmp36App

DAQRead.sln 2019-12-18 10:55
C:\...\DAQ CSharp Examples\DAQ CSharp USB-6008 Examples\DAQRead

AcqOneVoltageSample.2013.sln 2019-12-18 10:50
C:\...\Examples\DotNET4.5.1\Analog In\Measure Voltage\AcqOneVoltageSample\CS

GlobalContinuousAI_USB.sln 2019-12-18 10:27
C:\...\Examples\DotNET\v4.5\Applications\DAQmxWithUI\GlobalContinuousAI_USB\cs

Yesterday

TC01 DAQ Read.sln 2019-12-17 15:49
C:\...\DAQ CSharp TC-01 Thermocouple Examples\TC01 DAQ Read

DAQWrite.sln 2019-12-17 14:34
C:\...\DAQ CSharp Examples\DAQ CSharp USB-6008 Examples\DAQWrite

Get started



Clone or check out code

Get code from an online repository like GitHub or Azure DevOps



Open a project or solution

Open a local Visual Studio project or .sln file



Open a local folder

Navigate and edit code within any folder



Create a new project

Choose a project template with code scaffolding to get started

[Continue without code →](#)

Create a new project

Recent project templates

 Windows Forms App (.NET Framework)

C#

 ASP.NET Core Web Application

C#

 ASP.NET Web Application (.NET Framework)

C#

 ASP.NET Web Application (.NET Framework)

Visual Basic

 Windows Forms App (.NET Core)

C#

 Python Application

Python

Search for templates (Alt+S) 

[Clear all](#)

C#

Windows

Desktop

 NUnit Test Project (.NET Core)

A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.

C# Linux macOS Windows Desktop Test Web

 Windows Forms App (.NET Framework)

A project for creating an application with a Windows Forms (WinForms) user interface

C# Windows Desktop

 WPF App (.NET Framework)

Windows Presentation Foundation client application

C# Windows Desktop

 WPF App (.NET Core)

Windows Presentation Foundation client application

C# Windows Desktop

 WPF Custom Control Library (.NET Core)

Windows Presentation Foundation custom control library

C# Windows Desktop Library

 WPF User Control Library (.NET Core)

Windows Presentation Foundation user control library

C# Windows Desktop Library

 Blank App (Universal Windows)

A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.

[Back](#)

[Next](#)

File Edit View Project Build Debug Format Test Analyze Tools Extensions Window Help Search (Ctrl+Q) ThermistorApp Live Share

Toolbox

Form1.cs Form1.cs [Design]

Temperature Value: Get Data

Solution Explorer

Search Solution Explorer (Ctrl+Shift+F)

Solution 'ThermistorApp' (1 of 1 project)

ThermistorApp

- Properties
- References
- App.config
- Form1.cs
 - Form1.Designer.cs
 - Form1.resx
- Program.cs

Properties

Form1 System.Windows.Forms.Form

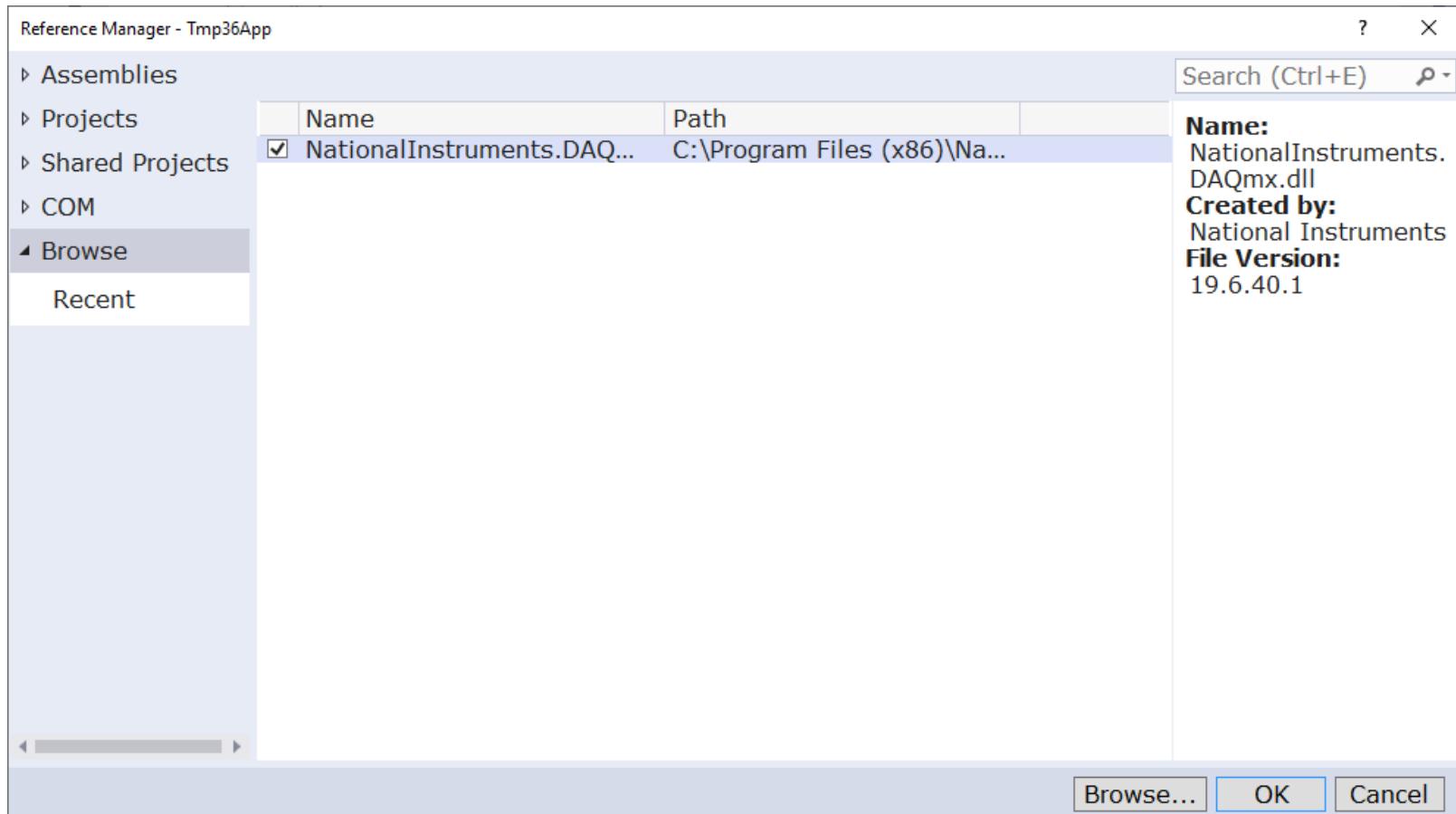
| | |
|-------------------|------------------------|
| Padding | 0, 0, 0, 0 |
| RightToLeft | No |
| RightToLeftLayout | False |
| ShowIcon | True |
| ShowInTaskbar | True |
| Size | 391, 231 |
| SizeGripStyle | Auto |
| StartPosition | WindowsDefaultLocation |
| Tag | |
| Text | Form1 |
| TopMost | False |

The text associated with the control.

Ready

15 , 15 391 x 231 Add to Source Control

NationalInstruments.DAQmx.dll



File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) ThermistorApp Live Share

Toolbox Form1.cs [Design] ThermistorApp ThermistorApp.Form1 btnGetTemperatureData_Click(object sender, EventArgs e)

General There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

```
11 } InitializeComponent();  
12 }  
13  
14 private void btnGetTemperatureData_Click(object sender, EventArgs e)  
15 {  
16     Task analogInTask = new Task();  
17  
18     AIChannel myAIChannel;  
19  
20     myAIChannel = analogInTask.AIChannels.CreateVoltageChannel(  
21         "dev1/ai0",  
22         "myAIChannel",  
23         AITerminalConfiguration.Rse,  
24         0,  
25         5,  
26         AIVoltageUnits.Volts  
27     );  
28  
29     AnalogSingleChannelReader reader = new AnalogSingleChannelReader(analogInTask.Stream);  
30  
31     double Vout = reader.ReadSingleSample();  
32  
33     double Vin = 5;  
34     double Ro = 10000; // 10k Resistor  
35     double Rt = (Vout * Ro) / (Vin - Vout);  
36     //Rt = 10000; //Used for Testing. Setting Rt = 10k should give TempC = 25  
37  
38     //Steinhart Constants  
39     double A = 0.001129148;  
40     double B = 0.000234125;  
41     double C = 0.0000000876741;  
42  
43     //Steinhart - Hart Equation  
44     double TempK = 1 / (A + (B * Math.Log(Rt)) + C * Math.Pow(Math.Log(Rt),3));  
45  
46     //Convert from Kelvin to Celsius  
47     double thermistorTempC = TempK - 273.15;  
48  
49     txtTempData.Text = thermistorTempC.ToString("0.00");  
50 }  
51 }  
52 }  
53 }
```

110 % No issues found Ln: 23 Ch: 45 SPC CRLF

Solution Explorer Solution 'ThermistorApp' (1 of 1 project) ThermistorApp Properties References App.config Form1.cs Form1.Designer.cs Form1.resx Program.cs

Properties

Add to Source Control

```

using System;
using System.Windows.Forms;
using NationalInstruments.DAQmx;

namespace ThermistorApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnGetTemperatureData_Click(object sender, EventArgs e)
        {
            Task analogInTask = new Task();

            AIChannel myAIChannel;

            myAIChannel = analogInTask.AIChannels.CreateVoltageChannel(
                "dev1/ai0",
                "myAIChannel",
                AITerminalConfiguration.Rse,
                0,
                5,
                AIVoltageUnits.Volts
            );

            AnalogSingleChannelReader reader = new AnalogSingleChannelReader(analogInTask.Stream);

            double Vout = reader.ReadSingleSample();

            double Vin = 5;
            double Ro = 10000; // 10k Resistor
            double Rt = (Vout * Ro) / (Vin - Vout);
            //Rt = 10000; //Used for Testing. Setting Rt = 10k should give TempC = 25

            //Steinhart Constants
            double A = 0.001129148;
            double B = 0.000234125;
            double C = 0.0000000876741;

            //Steinhart - Hart Equation
            double TempK = 1 / (A + (B * Math.Log(Rt)) + C * Math.Pow(Math.Log(Rt),3));

            //Convert from Kelvin to Celsius
            double thermistorTempC = TempK - 273.15;

            txtTempData.Text = thermistorTempC.ToString("0.00");
        }
    }
}

```

Read from DAQ

```
Task analogInTask = new Task();  
  
AIChannel myAIChannel;  
  
myAIChannel = analogInTask.AIChannels.CreateVoltageChannel(  
    "dev1/ai0",  
    "myAIChannel",  
    AITerminalConfiguration.Rse,  
    0,  
    5,  
    AIVoltageUnits.Volts  
);  
  
AnalogSingleChannelReader reader = new  
    AnalogSingleChannelReader(analogInTask.Stream);  
  
double Vout = reader.ReadSingleSample();
```

Calculate Temperature Value in degrees Celsius

```
...  
  
double Vout = reader.ReadSingleSample();  
  
double Vin = 5;  
double Ro = 10000; // 10k Resistor  
double Rt = (Vout * Ro) / (Vin - Vout);  
//Rt = 10000; //Used for Testing. Setting Rt = 10k should give TempC = 25  
  
//Steinhart Constants  
double A = 0.001129148;  
double B = 0.000234125;  
double C = 0.0000000876741;  
  
//Steinhart - Hart Equation  
double TempK = 1 / (A + (B * Math.Log(Rt)) + C * Math.Pow(Math.Log(Rt),3) );  
  
//Convert from Kelvin to Celsius  
double thermistorTempC = TempK - 273.15;  
  
txtTempData.Text = thermistorTempC.ToString("0.00");  
...
```

Final Application

The screenshot shows a Windows application window titled "Form1". The window has a standard title bar with minimize, maximize, and close buttons. Inside the window, there is a label "Temperature Value:" followed by a text input field containing the value "26.61". To the right of the input field is a button labeled "Get Data". The "Get Data" button is currently highlighted with a blue border.

Improvements

- Create and use separate **Classes** and in general improve the C# code
- Use a **Timer** in order to read values at specific intervals
- Plot values in a **Chart**
- Save Data to a **Database**
- Save Data to a **Text File**
- etc.

Good luck with your Application

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: <https://www.halvorsen.blog>

