LabVIEW LINX and Arduino

LabVIEW + LabVIEW LINX Toolkit + Arduino

Hans-Petter Halvorsen

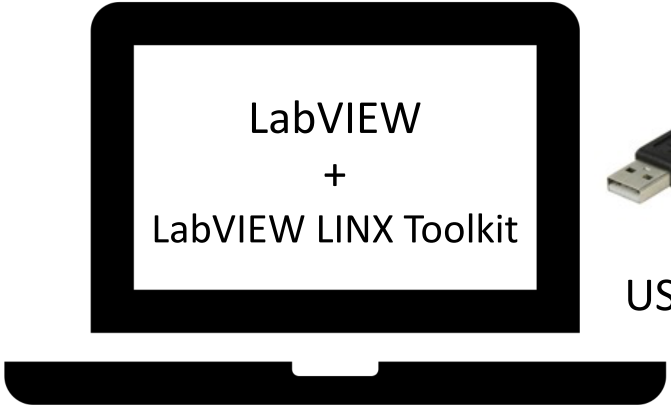
Contents

- This Tutorial shows how we can use Arduino in combination with the LabVIEW Programming environment
- “LabVIEW LINX Toolkit” is an add-on for LabVIEW which makes it possible to program the Arduino device using LabVIEW
- In that way we can create Data Logging Applications, etc. without the need of an expensive DAQ device
- If you don’t have “LabVIEW Professional” Software, you may use the “LabVIEW Community Edition” (free for non-commercial use). You then get a very low-cost DAQ/Datalogging System!

Table of Contents

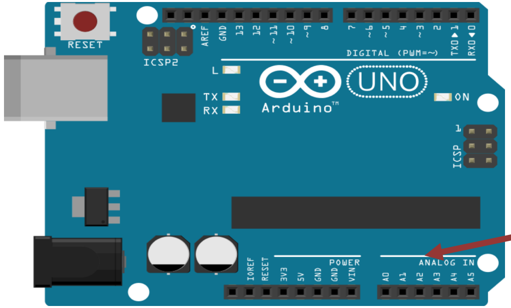
LabVIEW + LabVIEW LINX Toolkit

PC

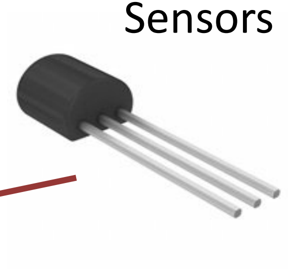


USB cable Type A-B

Arduino UNO



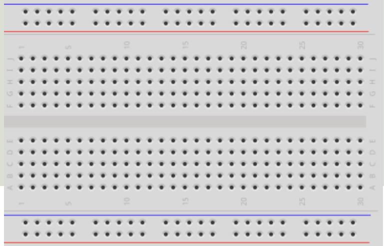
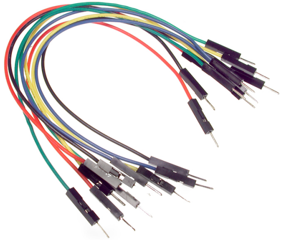
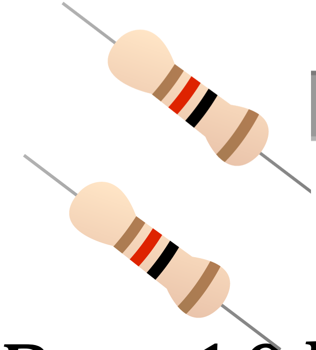
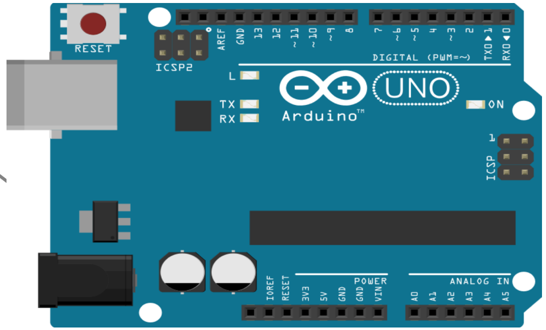
Sensors



TMP36
Temperature
Sensor

Hardware

- Arduino
- Breadboard
- Wires (Jumper Wires)
- Resistors ($R = 270\Omega, R = 10k\Omega$)
- LED, Push Button
- TMP36 Temperature Sensor
- Thermistor

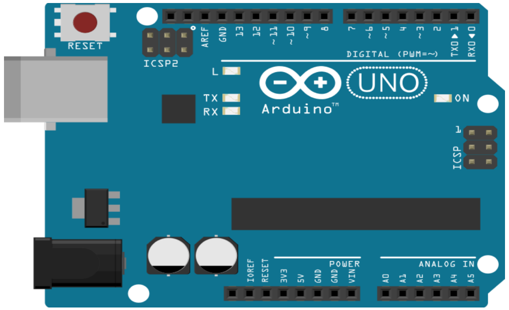








Arduino UNO

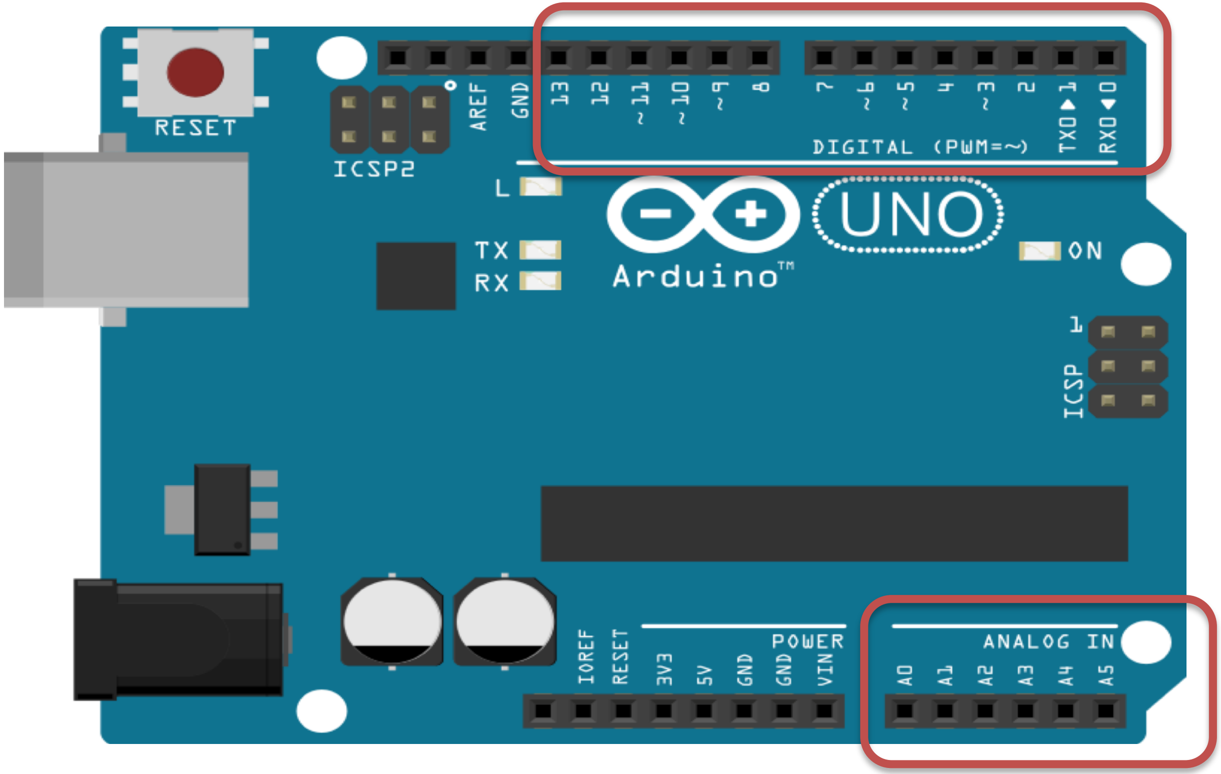
- Arduino is a Microcontroller
- Arduino is an open-source platform with Input/Output Pins (Digital In/Out, Analog In and PWM)
- Price about $20
- Arduino Starter Kit ~$40-80

with Cables, Wires, Resistors, Sensors, etc.



Arduino I/O Channels

Digital Inputs and Digital Outputs



You can choose from the code if they are to be inputs or outputs

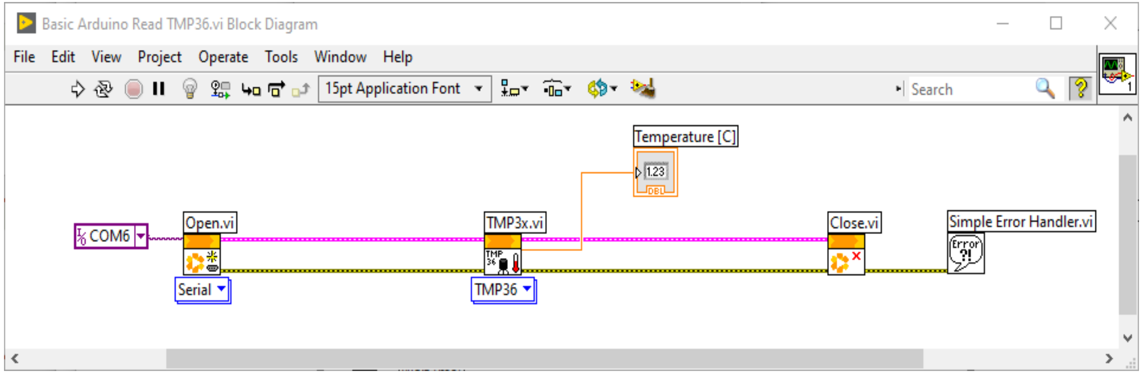
Those marked with ~ can also be used as "Analog Outputs", so-called PWM outputs

Analog Inputs

LabVIEW

- LabVIEW is Graphical Software
- LabVIEW has powerful features for simulation, control and DAQ applications

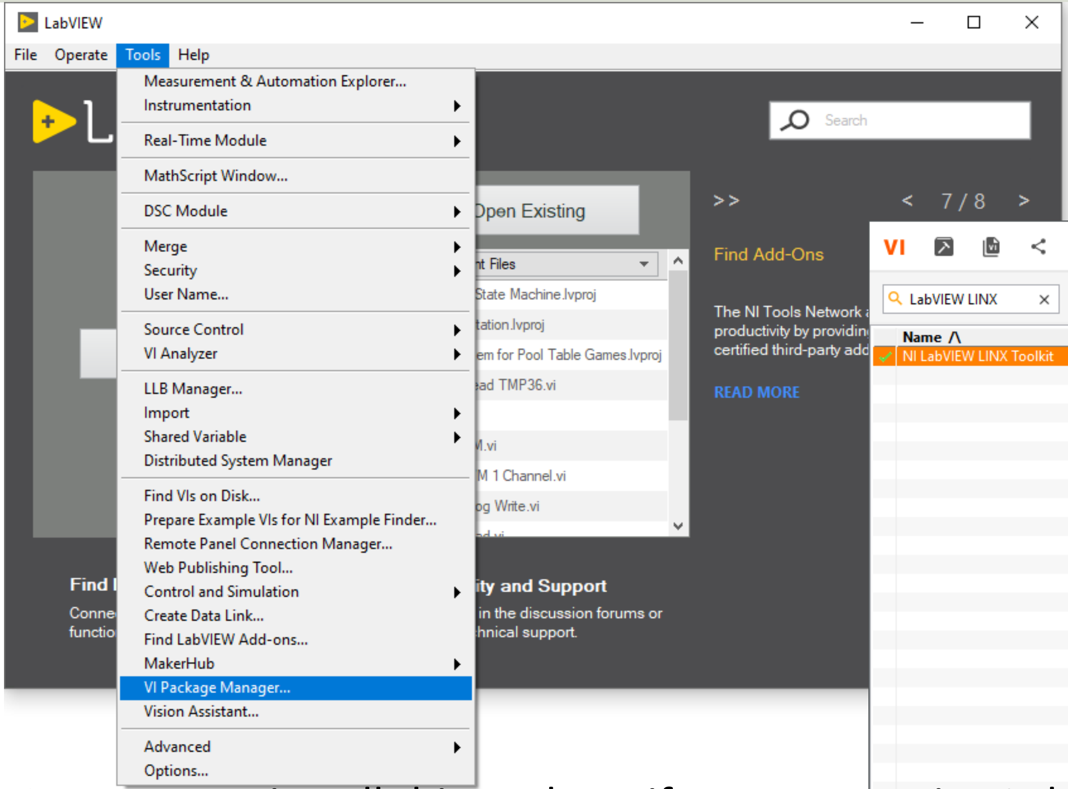
Basic LabVIEW Example:



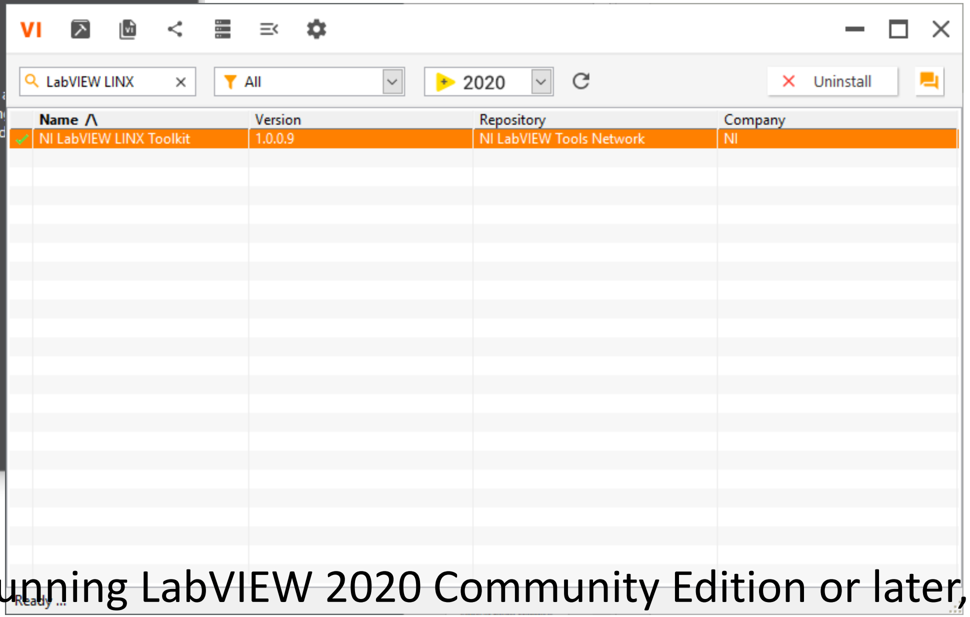
LabVIEW LINX Toolkit

- The LabVIEW LINX Toolkit adds support for Arduino, Raspberry Pi, and BeagleBone embedded platforms
- We will use Arduino Uno in this Tutorial

Installing LabVIEW LINX Toolkit

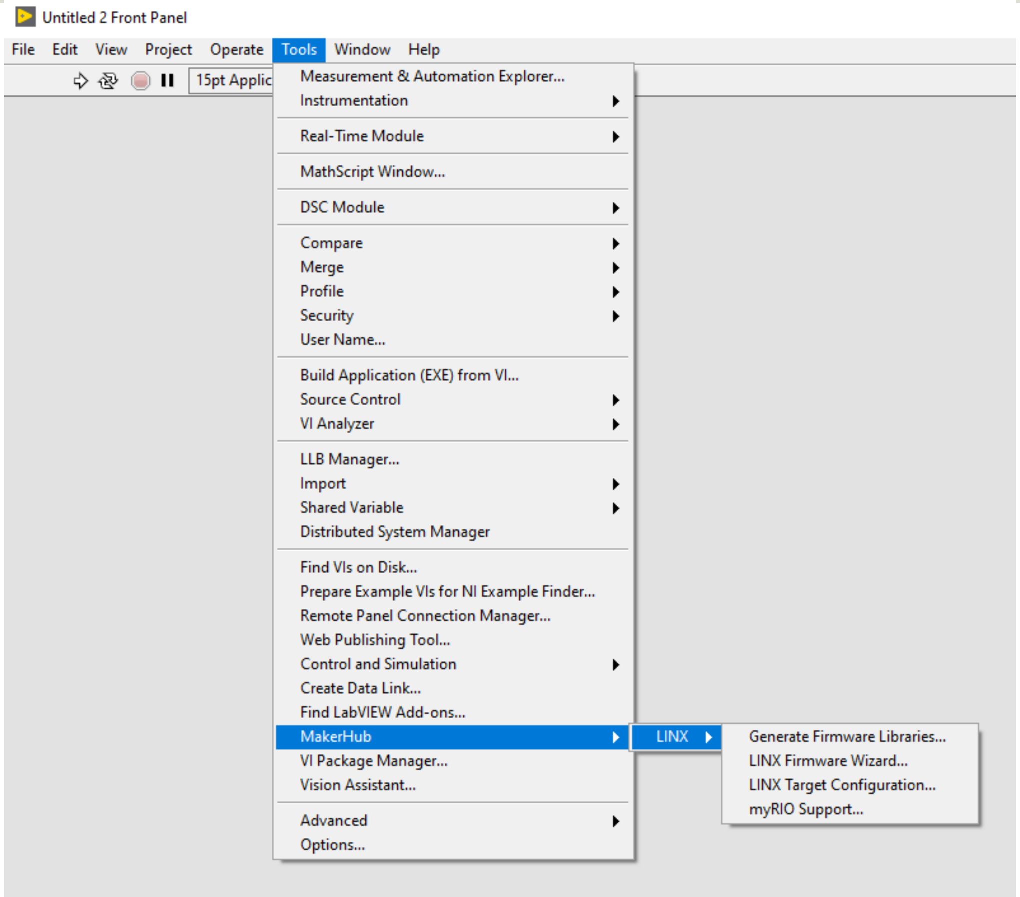


Use VI Package Manger

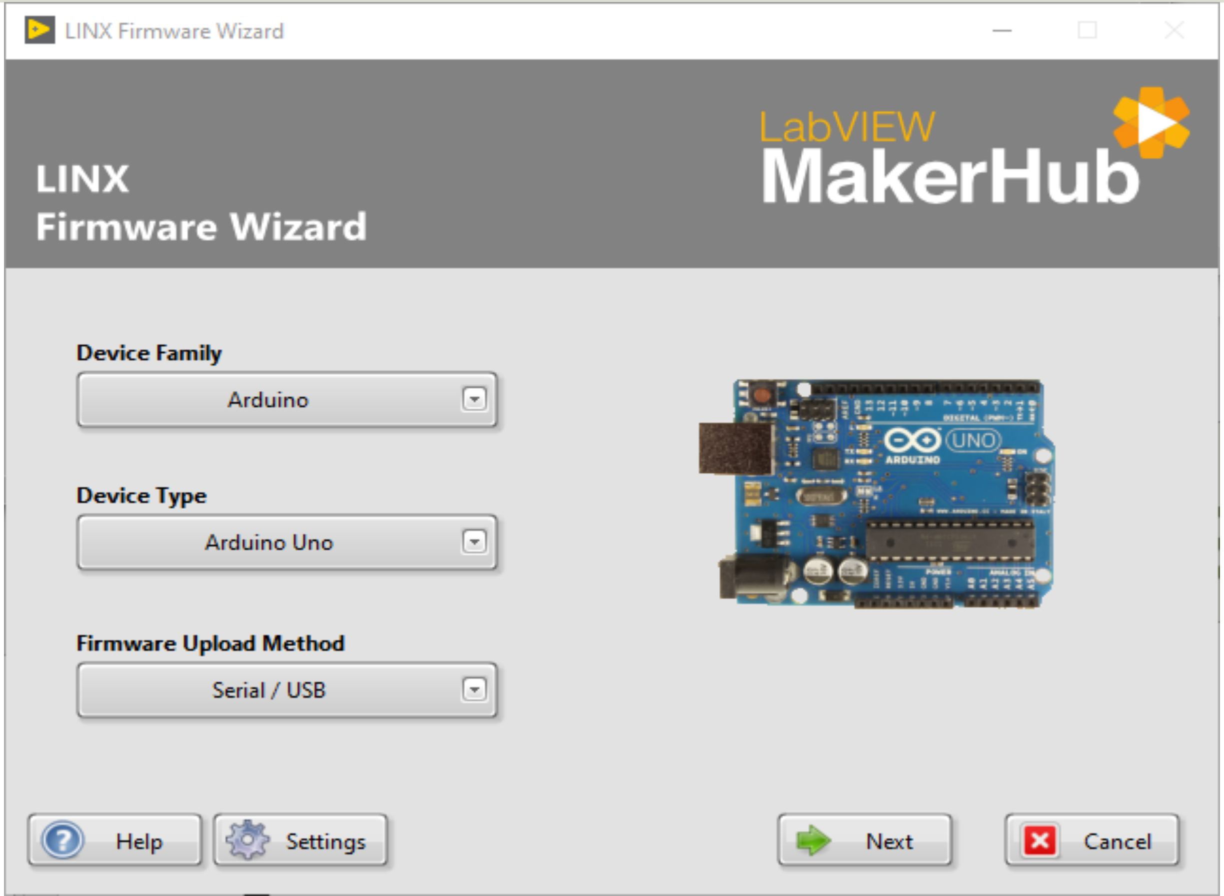


Note: Do not install this package if you are running LabVIEW 2020 Community Edition or later, as the Community Edition already includes the LabVIEW LINX Toolkit

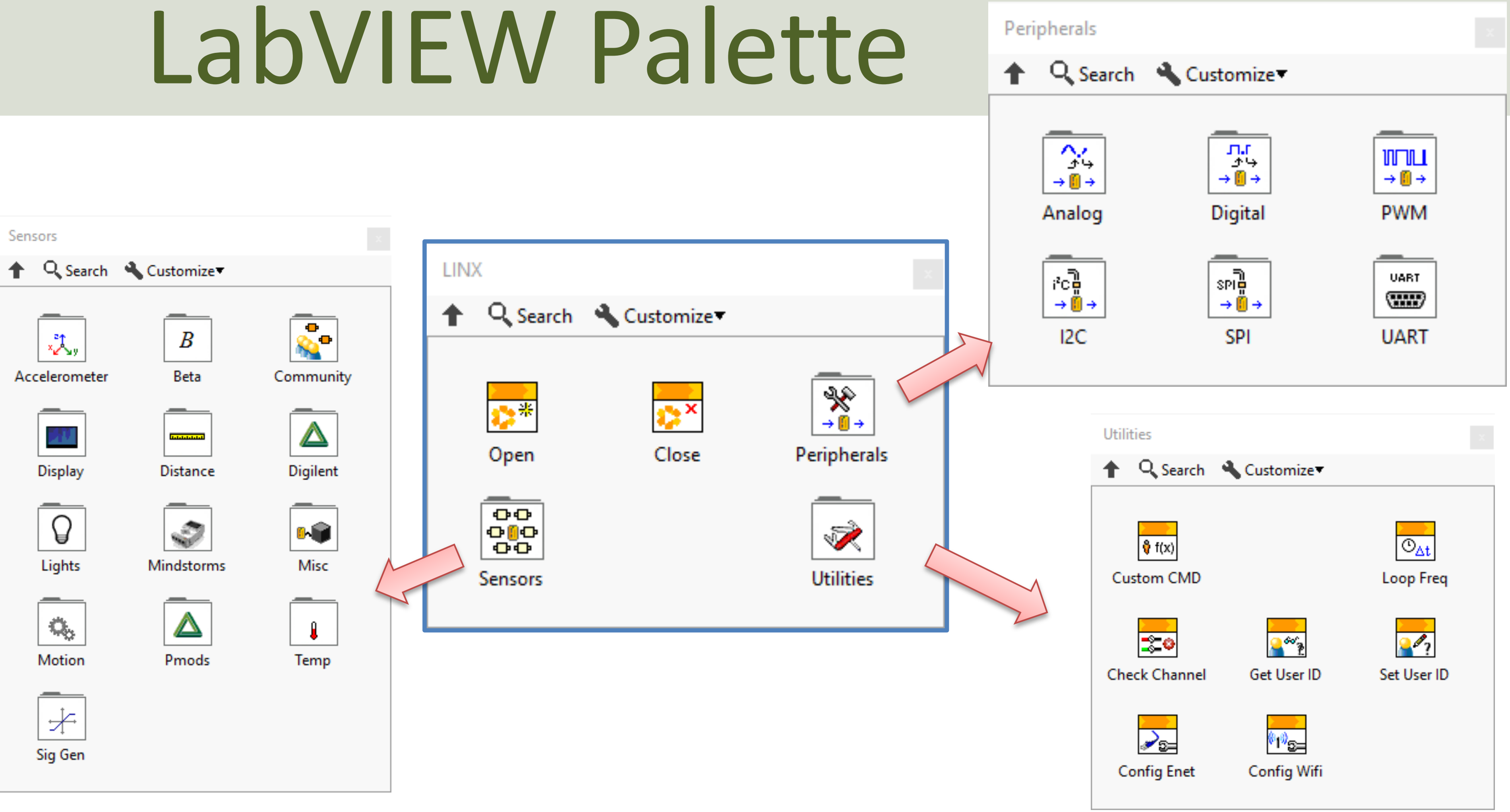
LabVIEW LINX



LINX Firmware Wizard



LabVIEW Palette



DAQ System

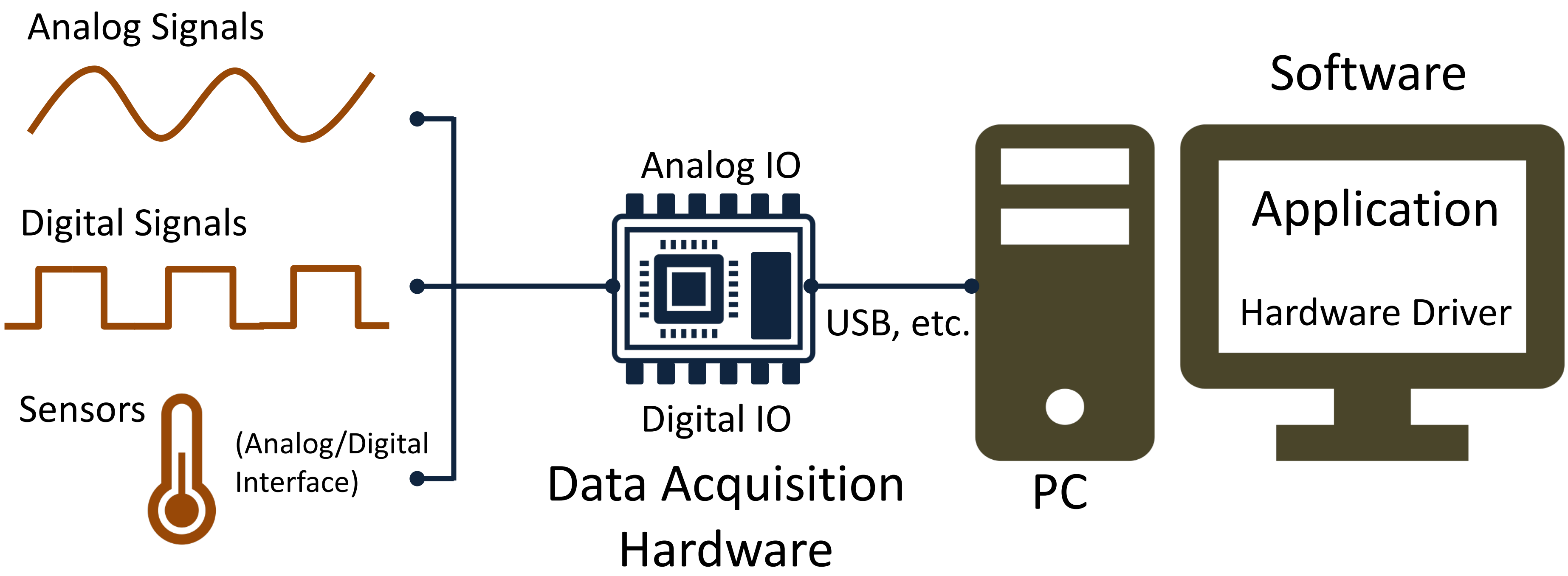
Hans-Petter Halvorsen

Table of Contents

DAQ System

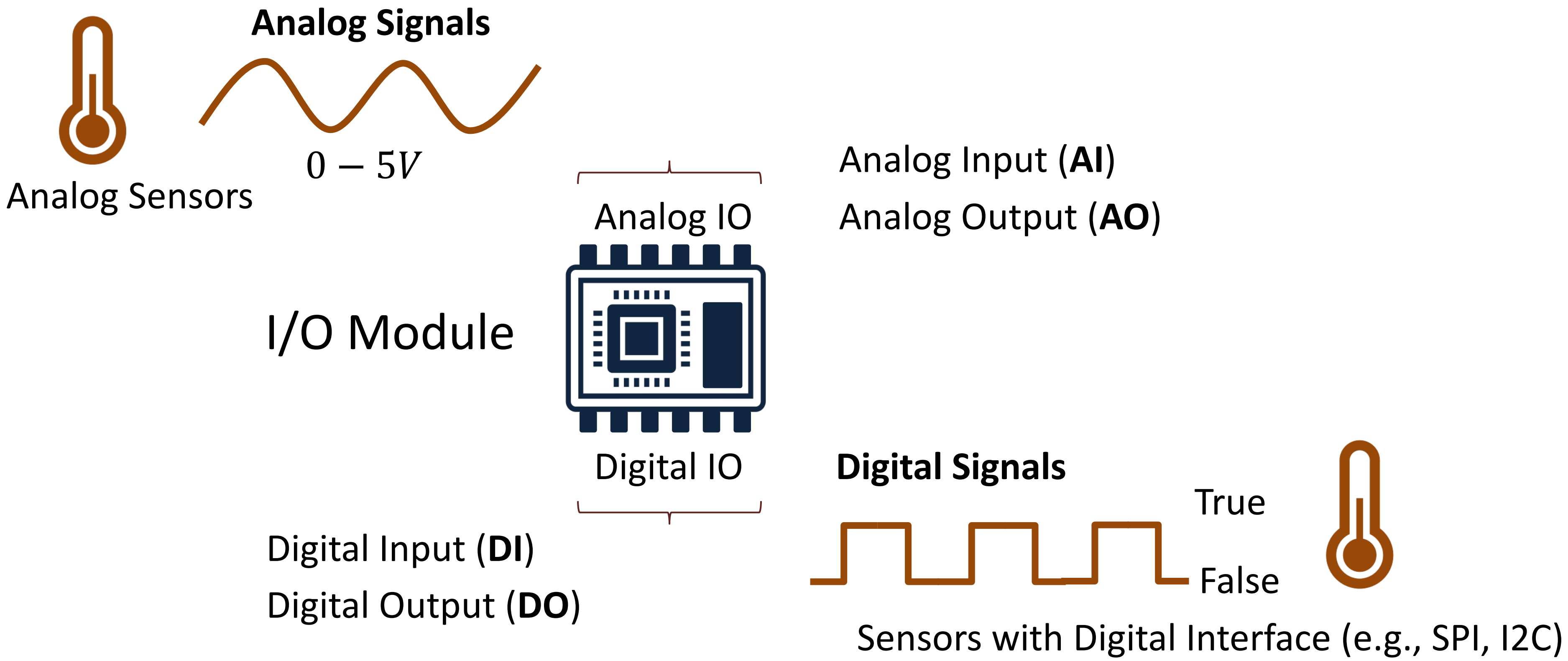
DAQ – Data Acquisition

Input/Output Signals



We will use an Arduino Uno as the DAQ Hardware

I/O Module


Analog Signals
Analog Sensors
0 – 5V
Analog IO
Analog Input (AI)
Analog Output (AO)
I/O Module
Digital IO
Digital Input (DI)
Digital Output (DO)
Digital Signals
True
False
Sensors with Digital Interface (e.g., SPI, I2C)

LabVIEW LINX Arduino DAQ System

Hans-Petter Halvorsen

Table of Contents

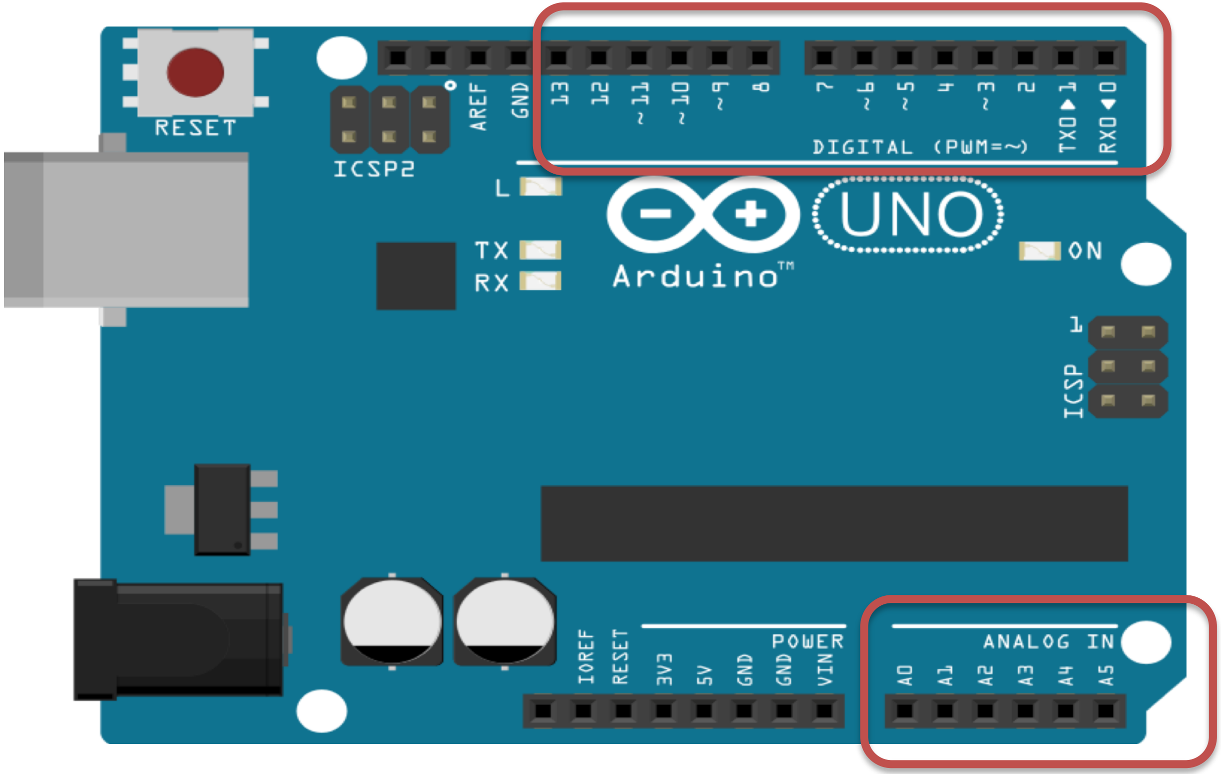
Input/Output Channels

Hans-Petter Halvorsen

Table of Contents

Arduino I/O Channels

Digital Inputs and Digital Outputs



You can choose from the code if they are to be inputs or outputs

Those marked with ~ can also be used as "Analog Outputs", so-called PWM outputs

Analog Inputs

Input/Output Channels

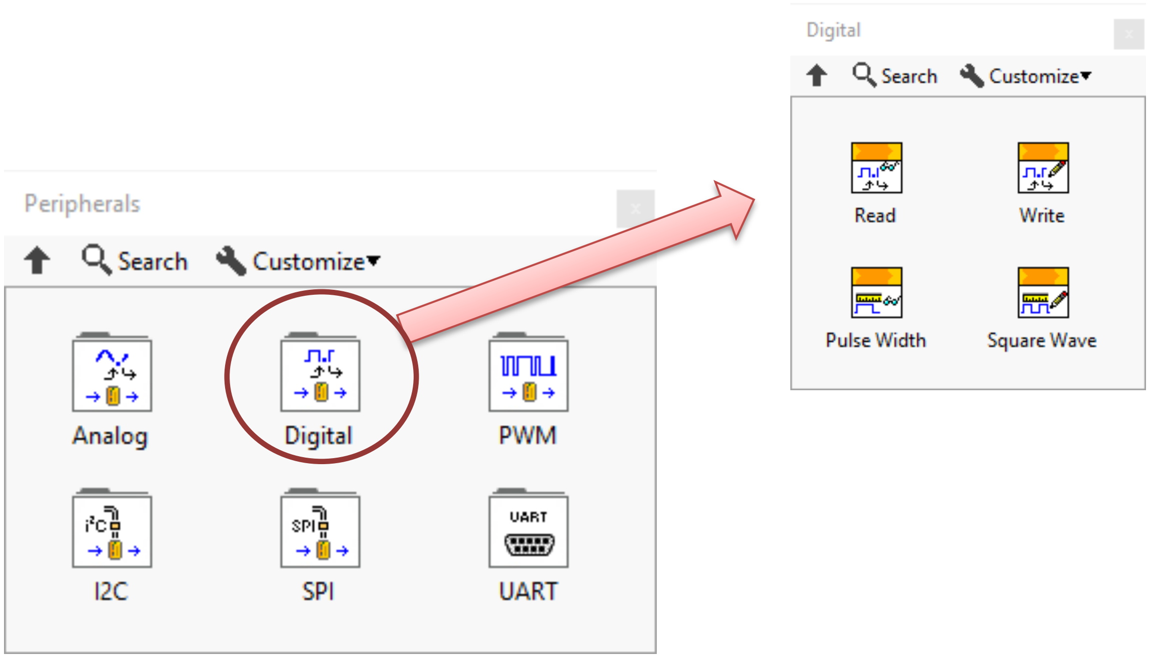
- Digital
  - Digital Out
  - Digital In
- Analog
  - Analog Out
  - Analog In

Digital I/O

Hans-Petter Halvorsen

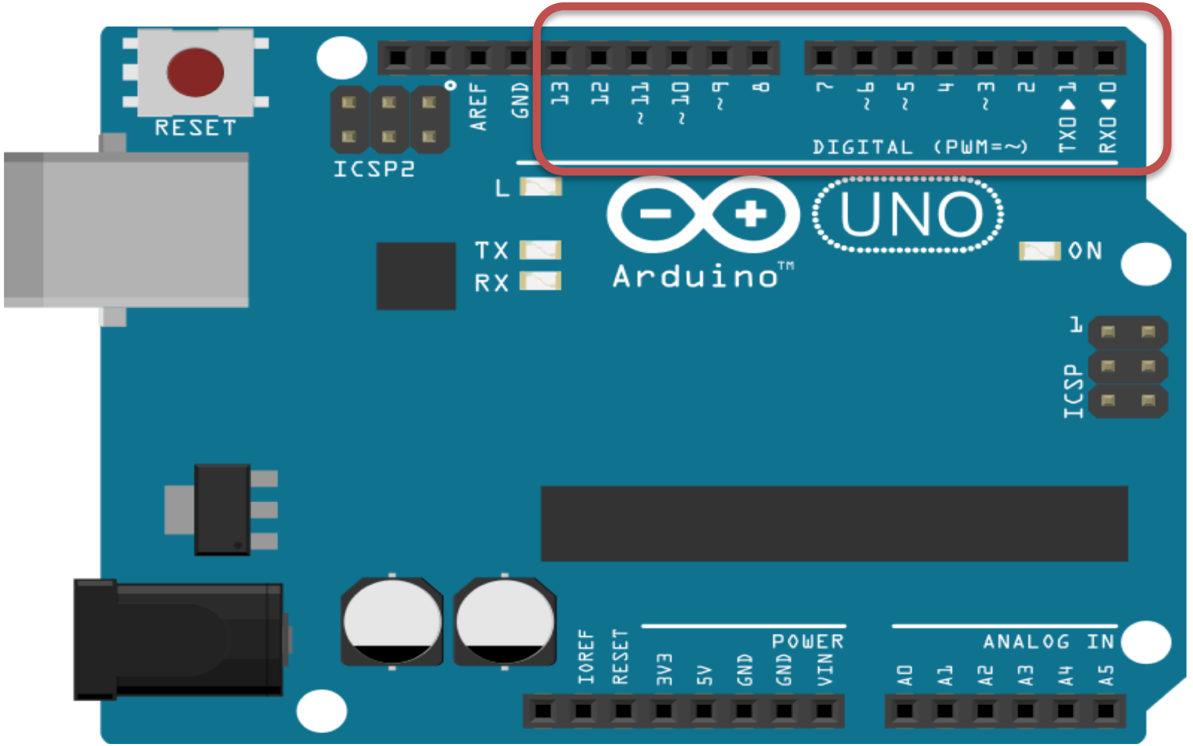
Table of Contents

LabVIEW Palette – Digital I/O



Digital I/O

Digital Inputs and Digital Outputs



You can choose from the code if they are to be inputs or outputs

Those marked with ~ can also be used as "Analog Outputs", so-called PWM outputs

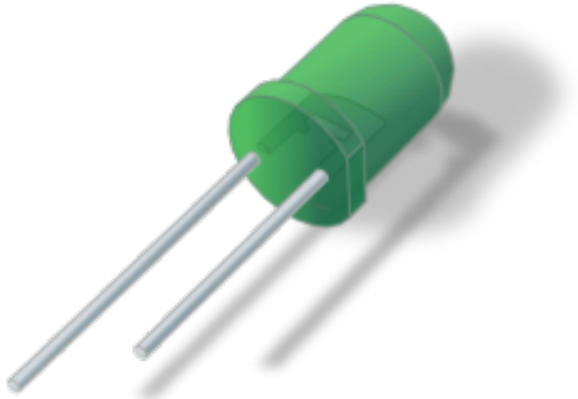
Digital Out (DO)

Hans-Petter Halvorsen

Table of Contents

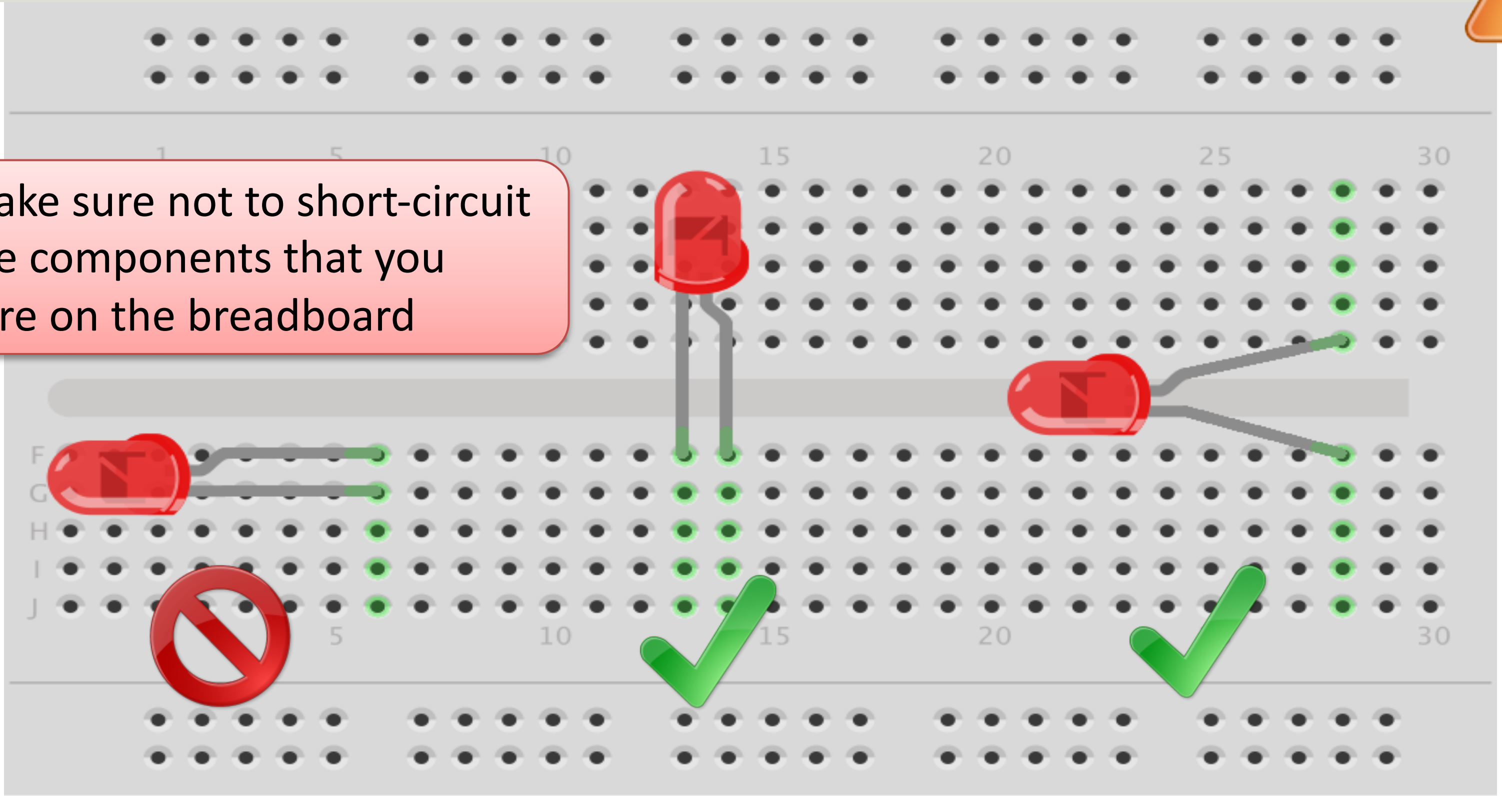
Digital Out (DO)

- We will use the Digital Out pins to turn on/off a LED



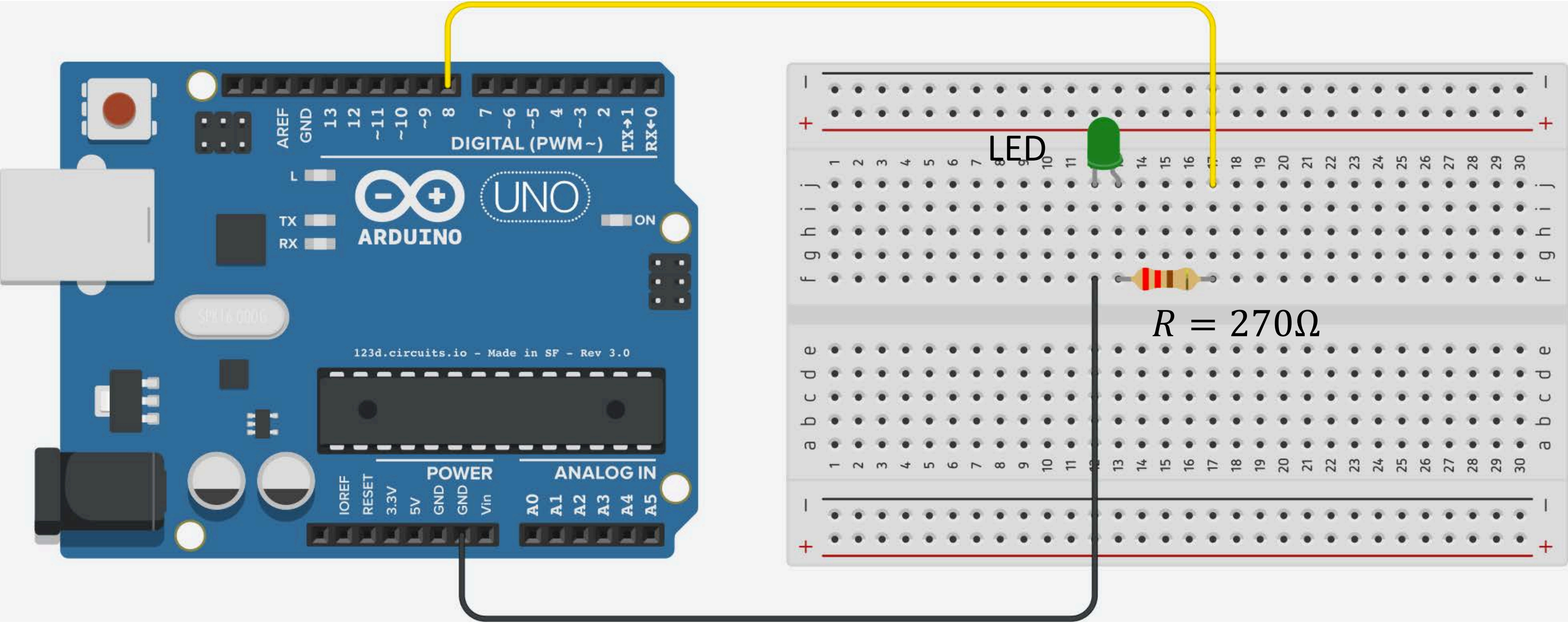
Breadboard Wiring

Make sure not to short-circuit the components that you wire on the breadboard



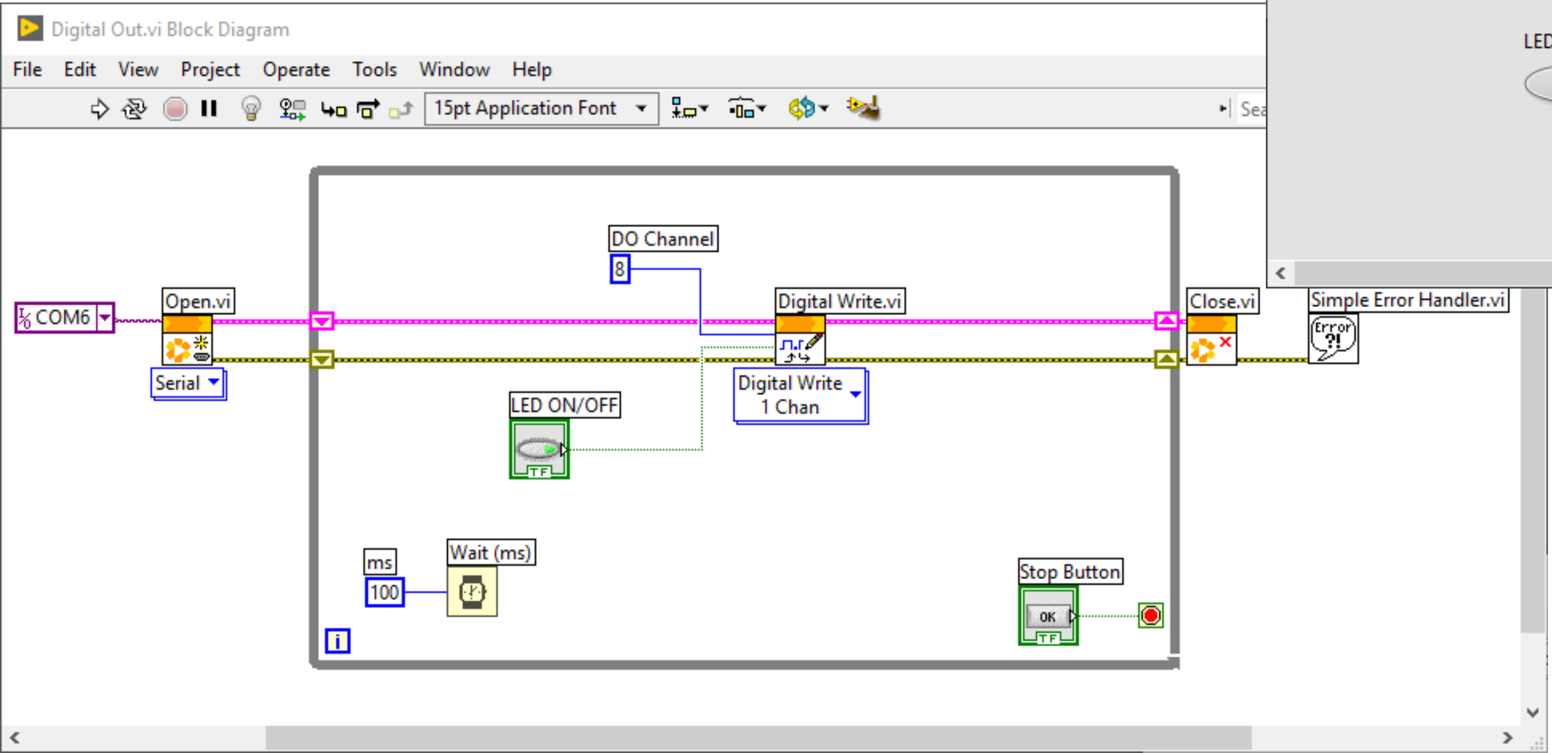
The Breadboard is used to connect components and electrical circuits

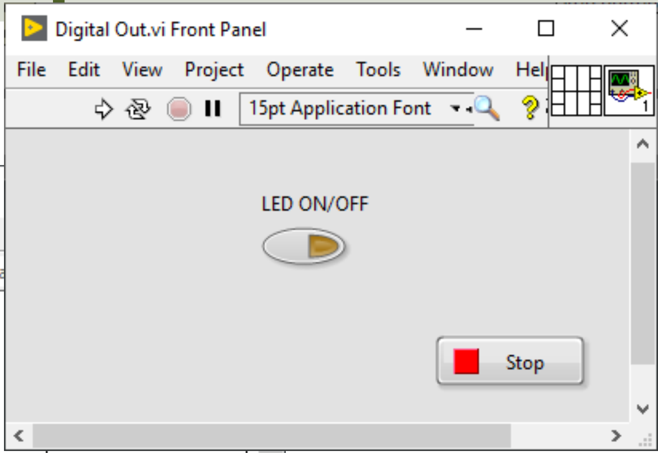
LED Wiring



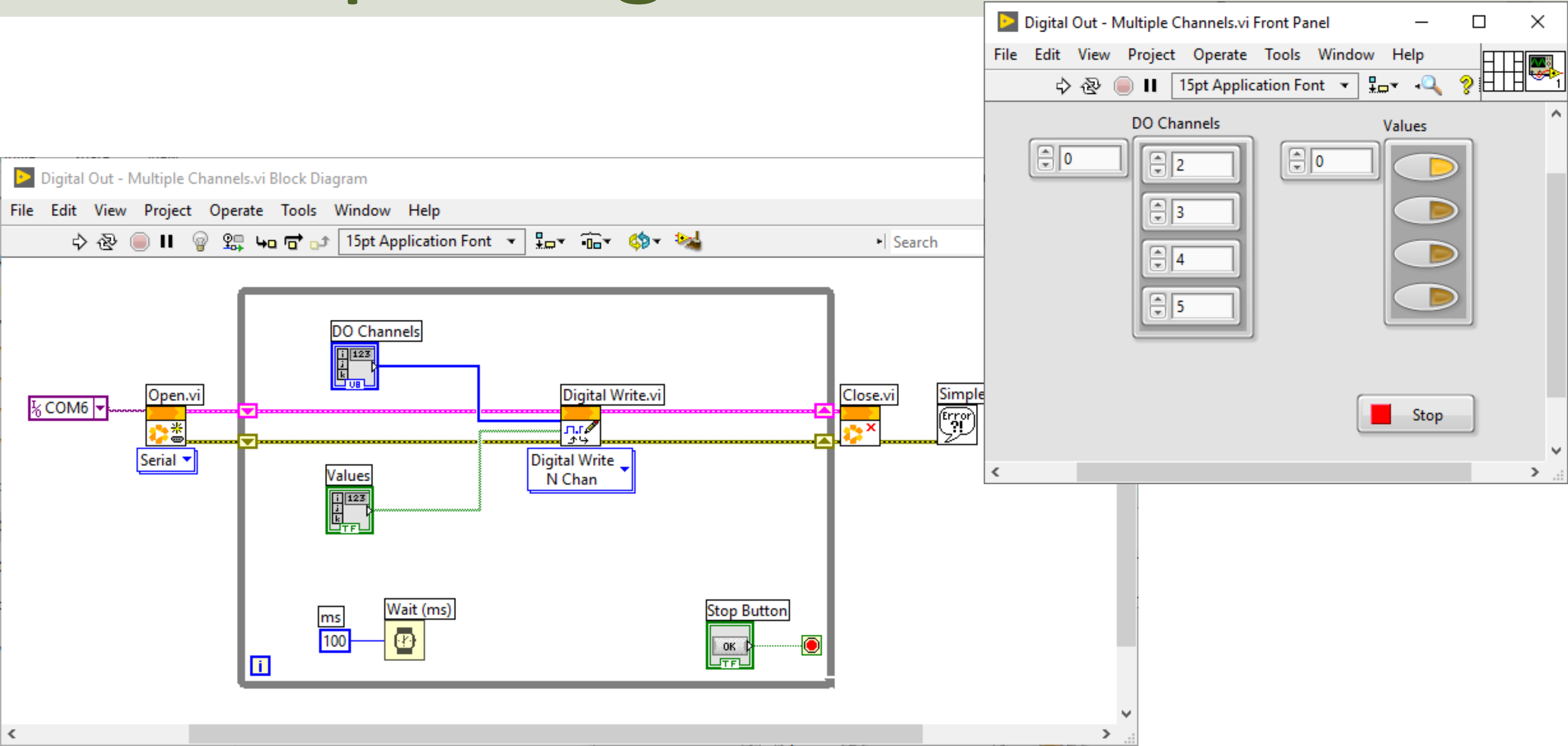
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
UNO
ARDUINO
ON
123d.circuits.io - Made in SF - Rev 3.0
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
LED
$R = 270\Omega$

LabVIEW Example





Multiple Digital Out Channels



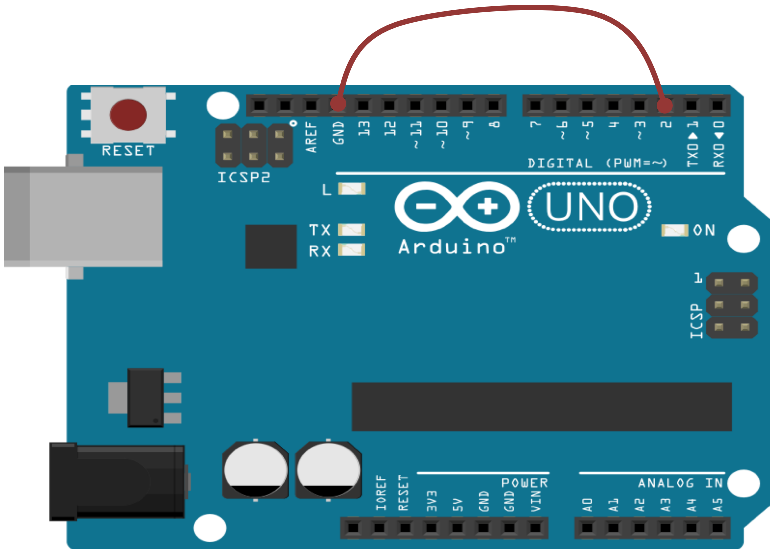
Digital In (DI)

Hans-Petter Halvorsen

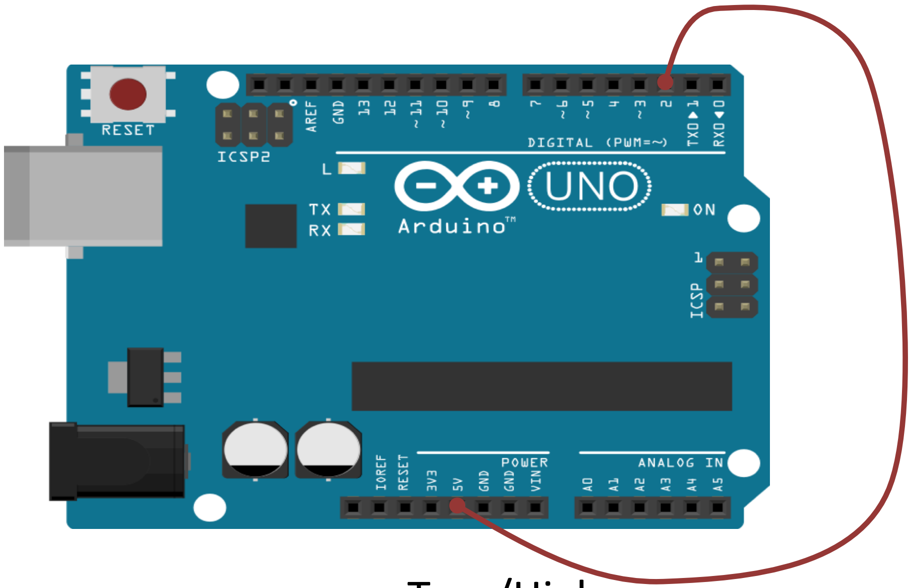
Table of Contents

LabVIEW - Digital In

We can test the Digital In (Read) by wiring to 5V (True/High) or GND (False/Low)

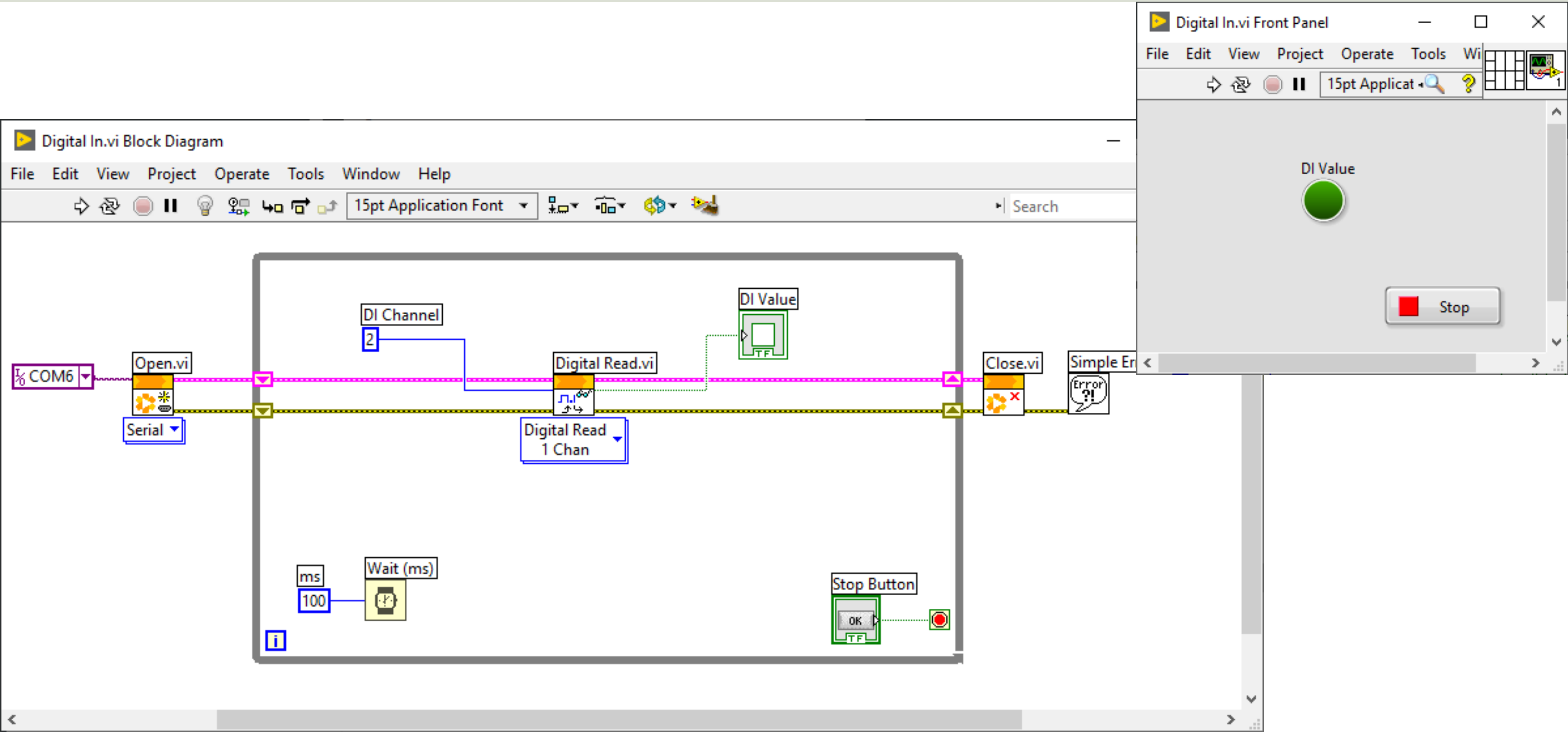


False/Low



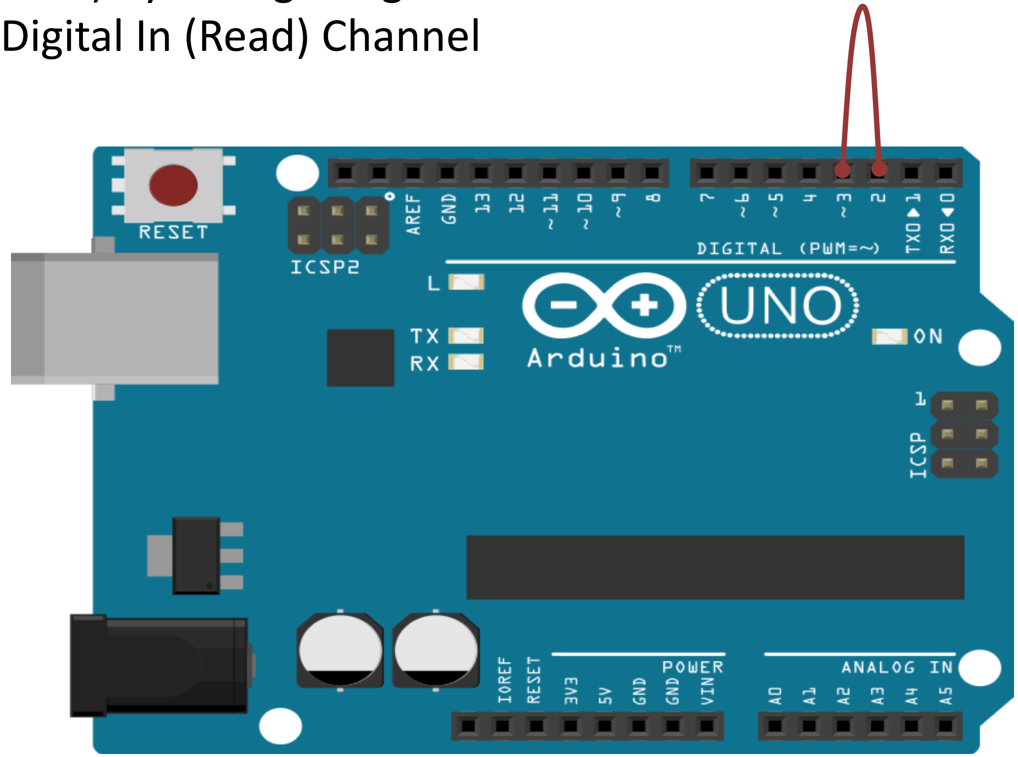
True/High

LabVIEW - Digital In

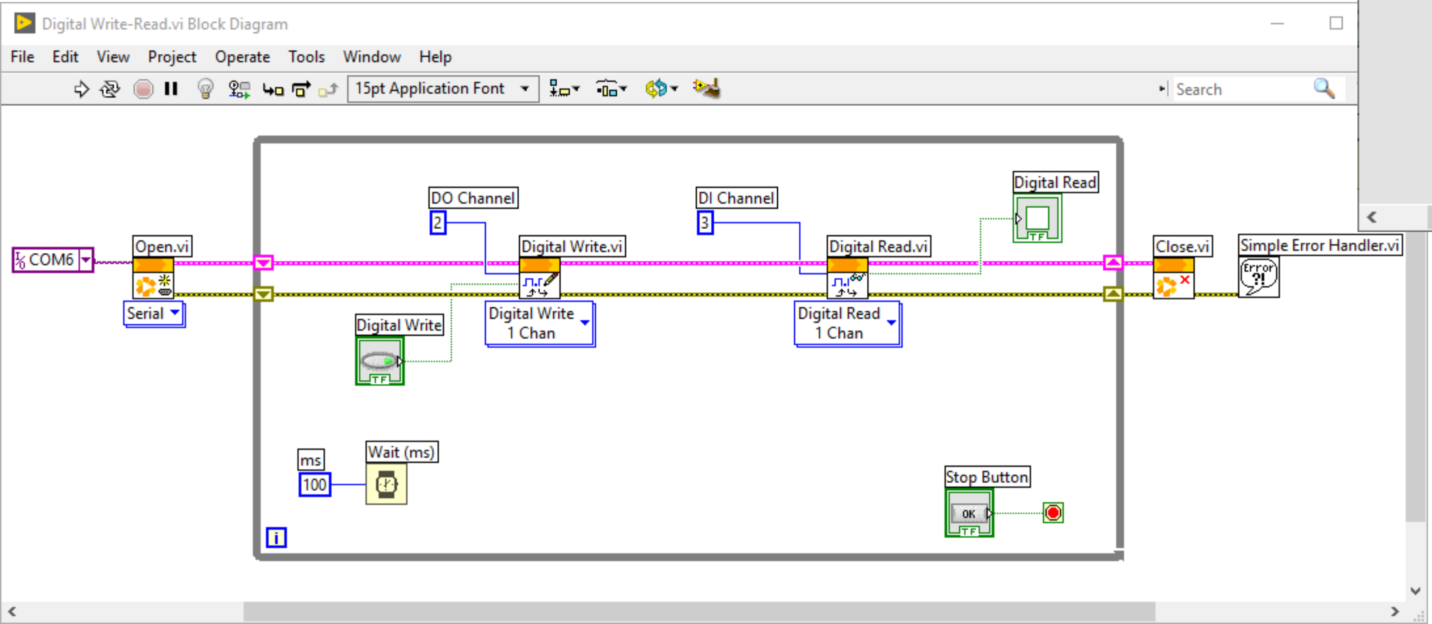


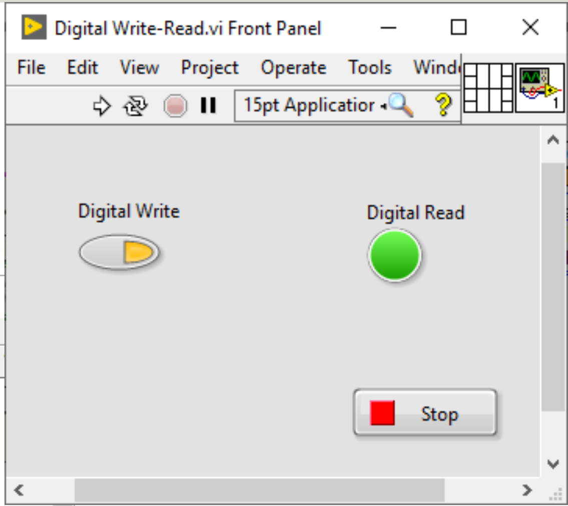
LabVIEW Digital Write - Read

We can test the Digital In (Read) by wiring a Digital Out (Read) Channel to the Digital In (Read) Channel



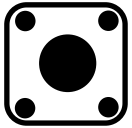
LabVIEW Digital Write - Read

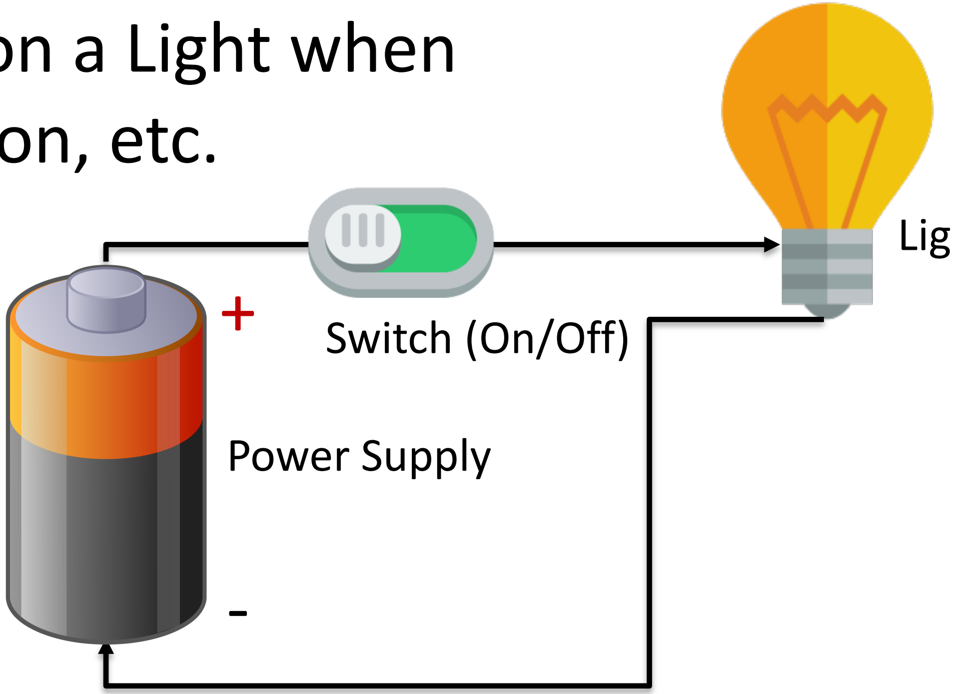




Push Button/Switch

- Pushbuttons or switches connect two points in a circuit when you press them.
- You can use it to turn on a Light when holding down the button, etc.



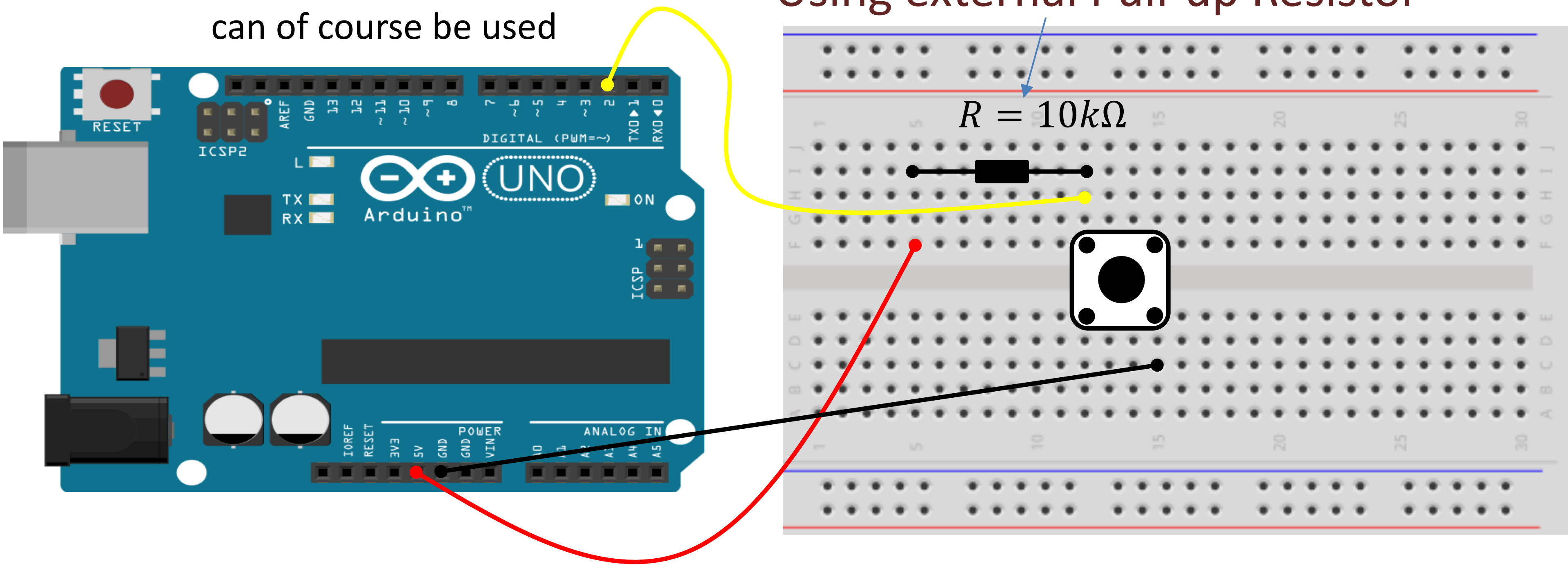


Wiring (Pull-up Resistor)

Another Digital Channel
can of course be used

Using external Pull-up Resistor

$R = 10k\Omega$

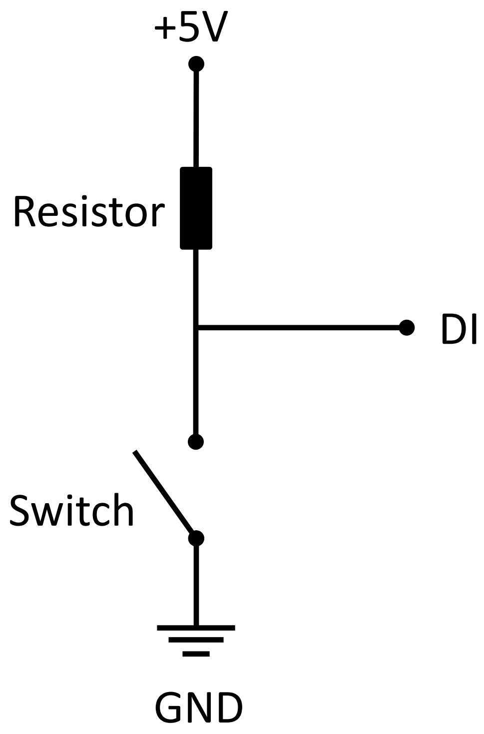


Pull-down/Pull-up Resistor

Why do we need a pull-up or pull-down resistor in the circuit?

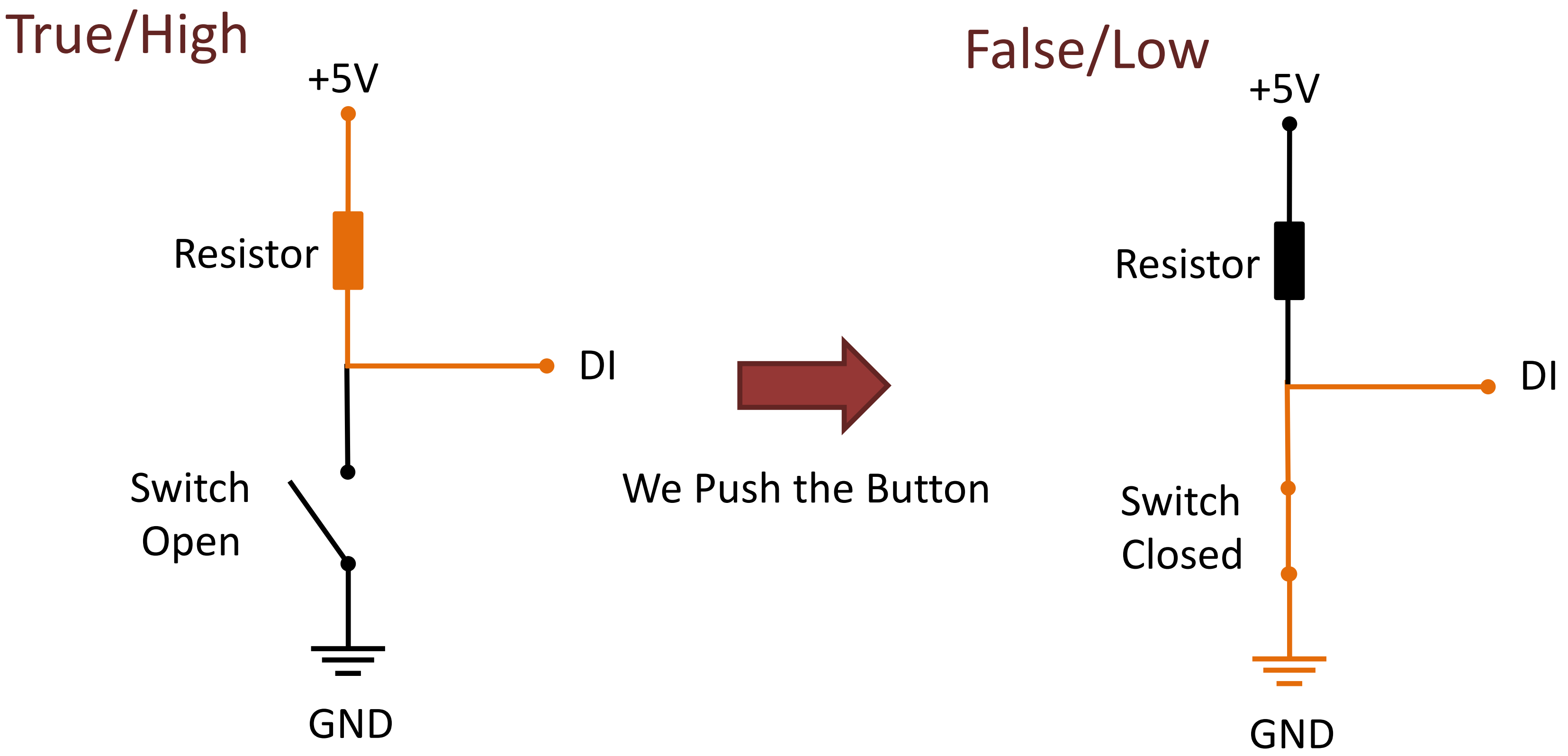
- If you disconnect the digital I/O pin from everything, it will behave in an irregular way.
- This is because the input is "floating" - that is, it will randomly return either HIGH or LOW.
- That's why you need a pull-up or pull-down resistor in the circuit.

Pull-up Resistor

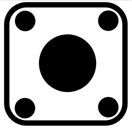


- When the pushbutton is open (unpressed) there is a connection between 5V and the DI pin.
- This means the default state is **True** (High).
- When the button is closed (pressed), the state goes to **False** (Low).

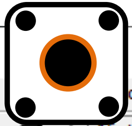
Pull-up Resistor



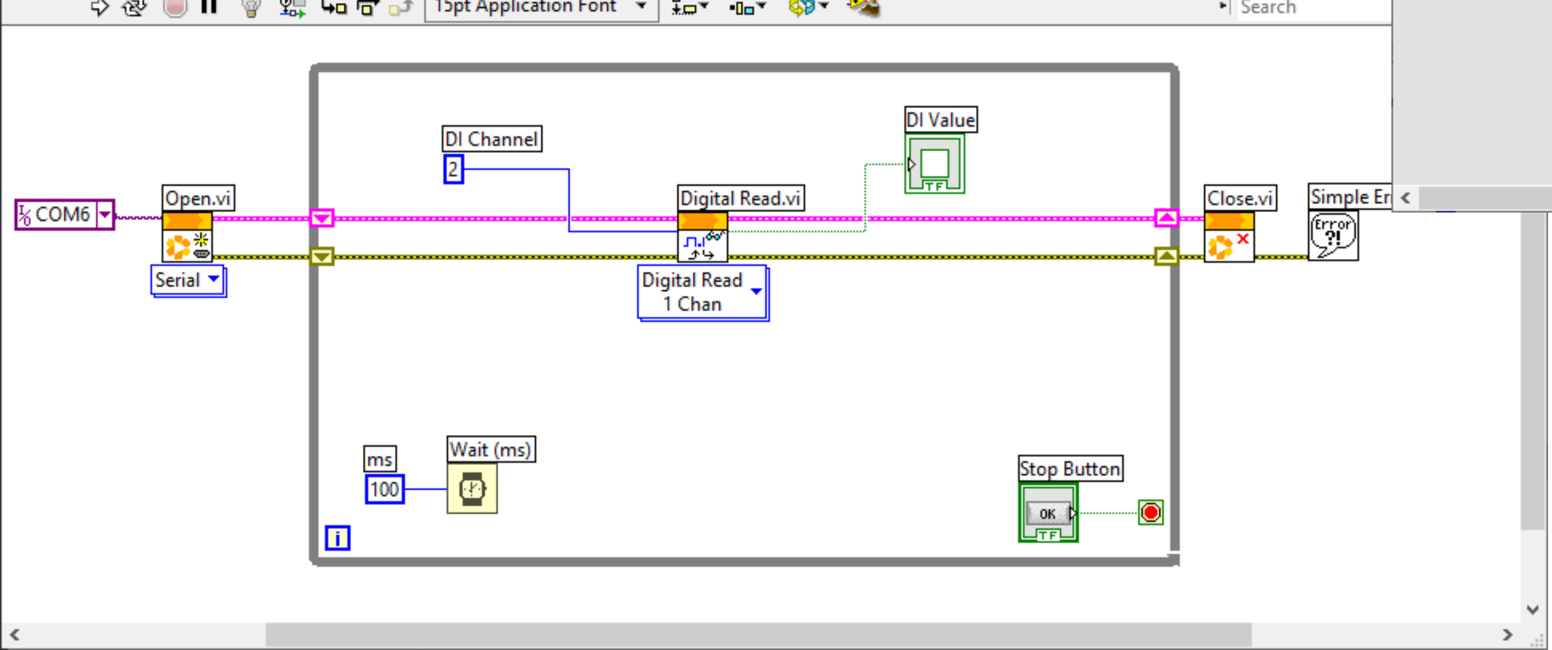
Push Button (Pull-up Resistor)

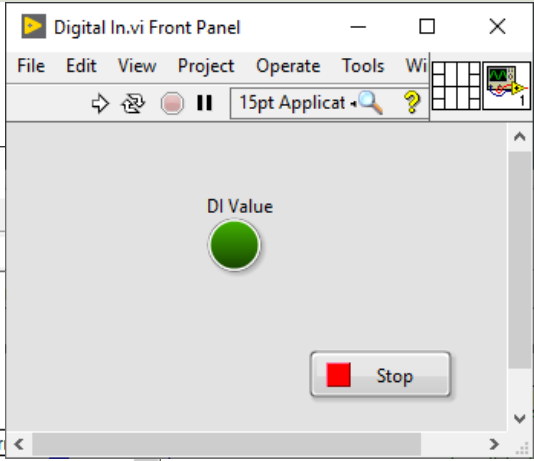


Button is NOT Pushed => True/High

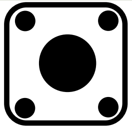


Button is Pushed => False/Low

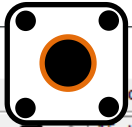




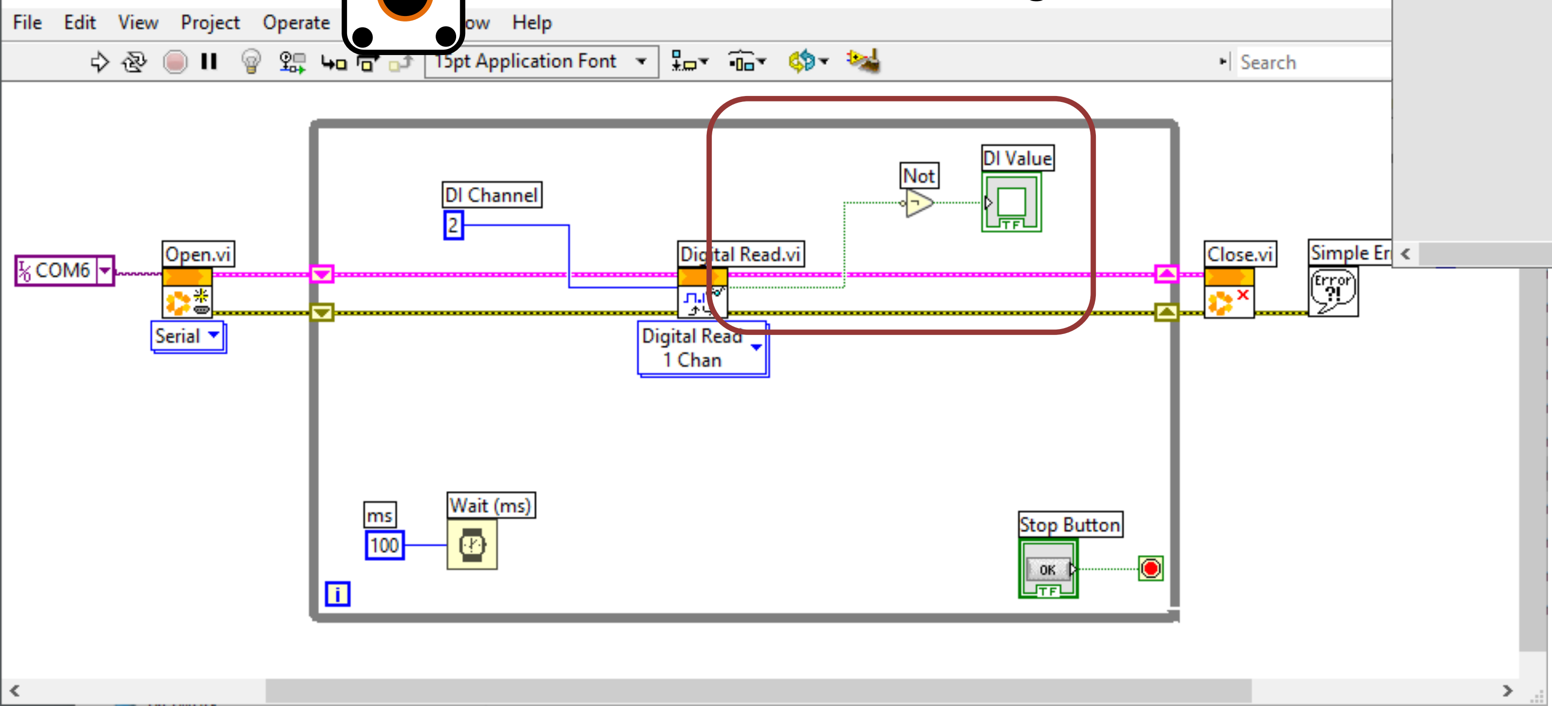
Adding a “NOT” block

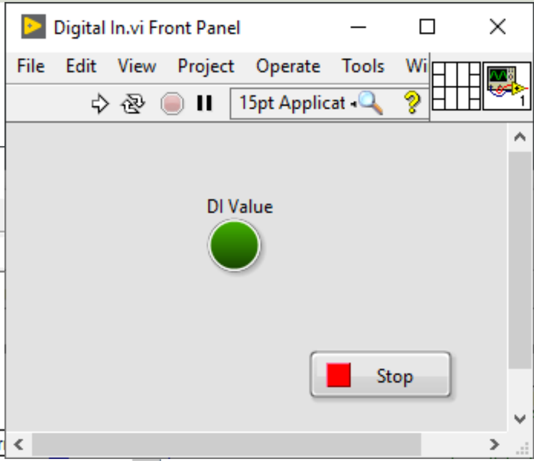


Button is NOT Pushed => False/Low



Button is Pushed => True/High

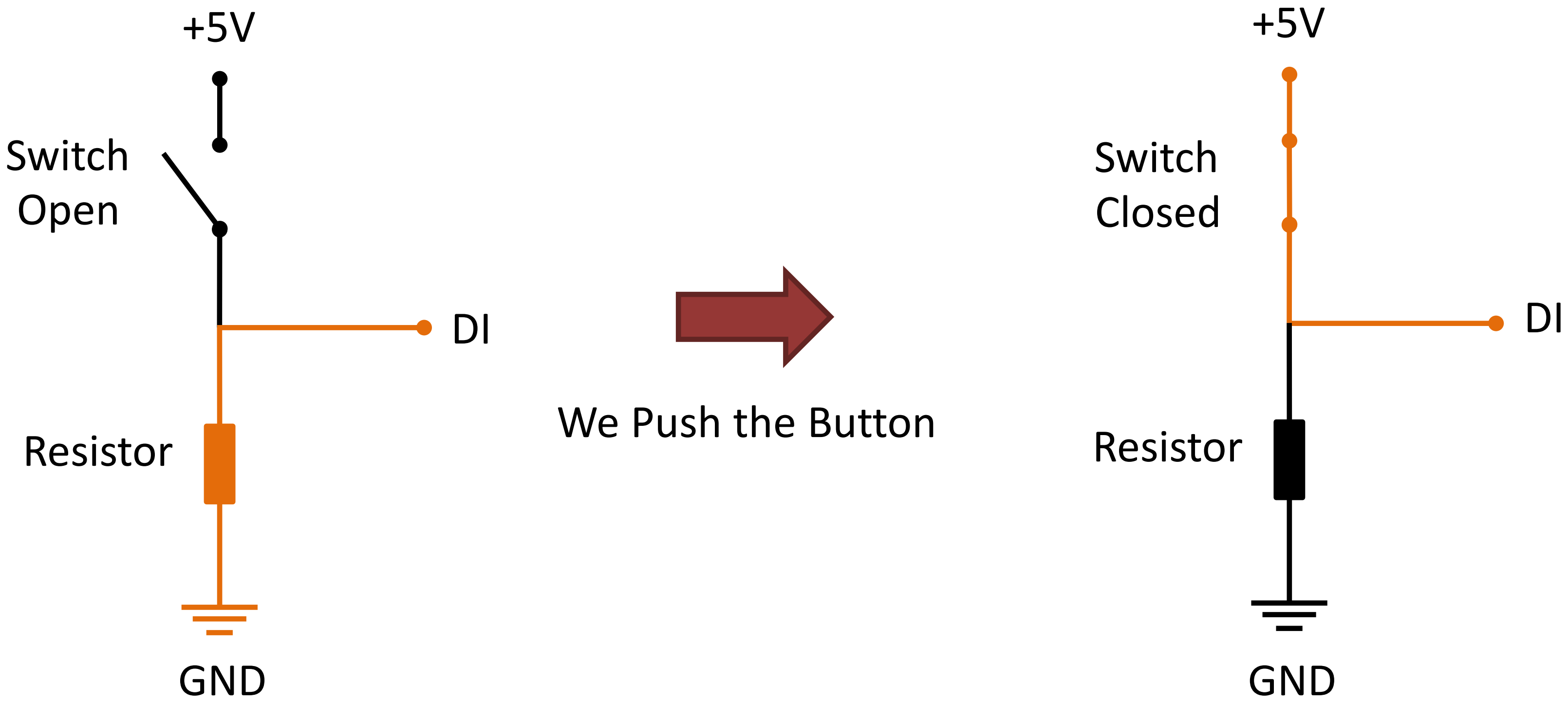




Pull-down Resistor

We could also have wired according to a “Pull-down“ Resistor

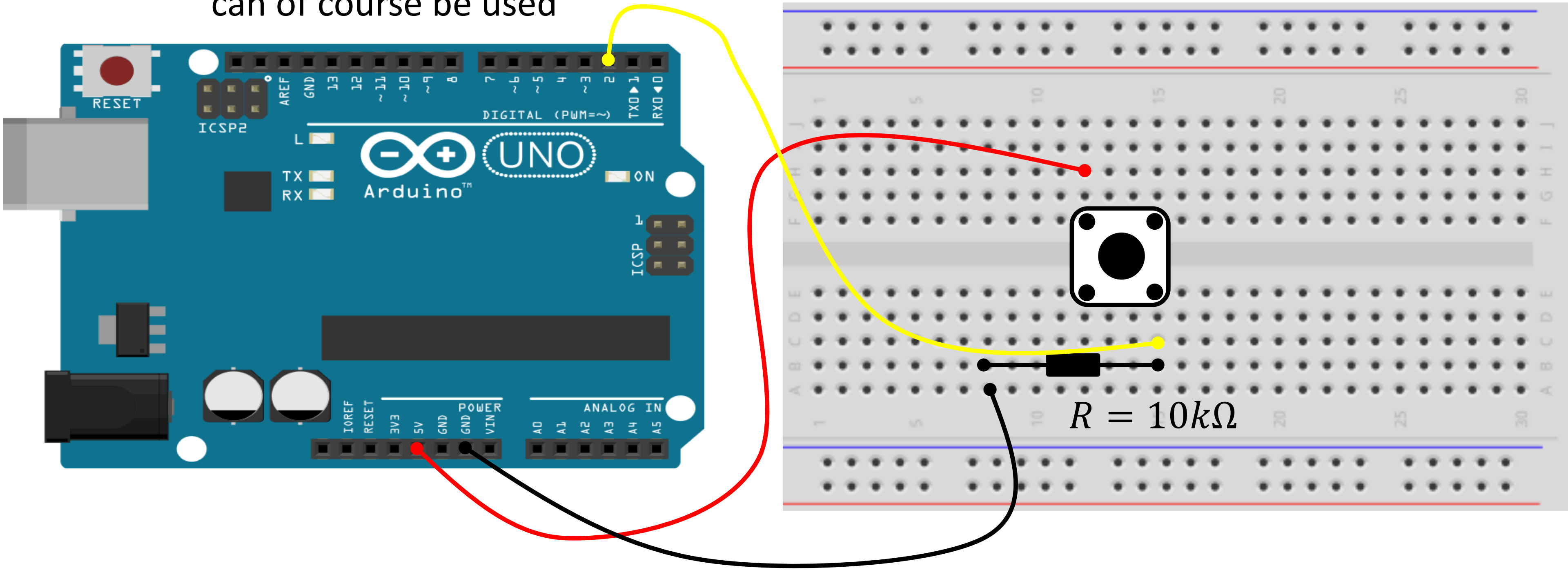
False/Low　　　　　　　　　　True/High



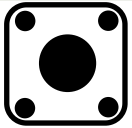
Wiring (Pull-down Resistor)

Another Digital Channel
can of course be used

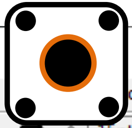
Using external Pull-down Resistor



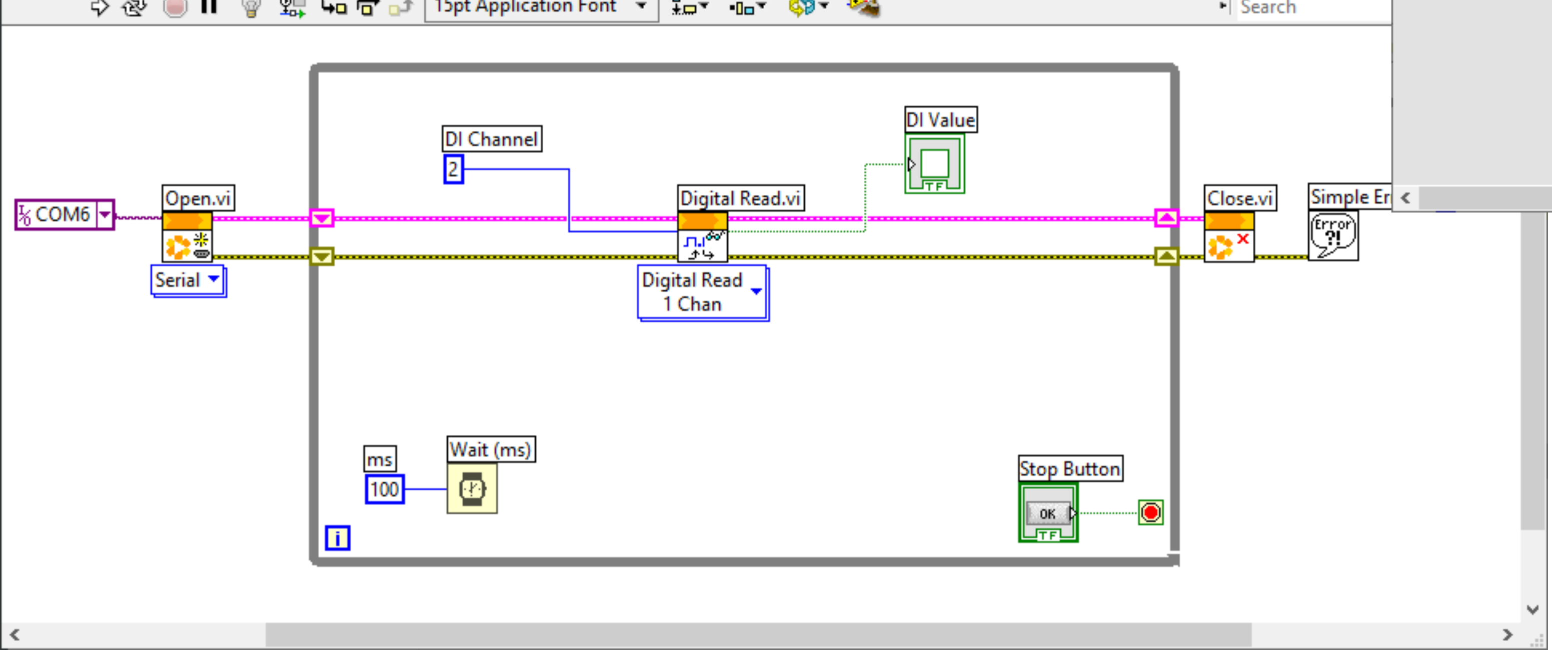
Push Button (Pull-down Resistor)

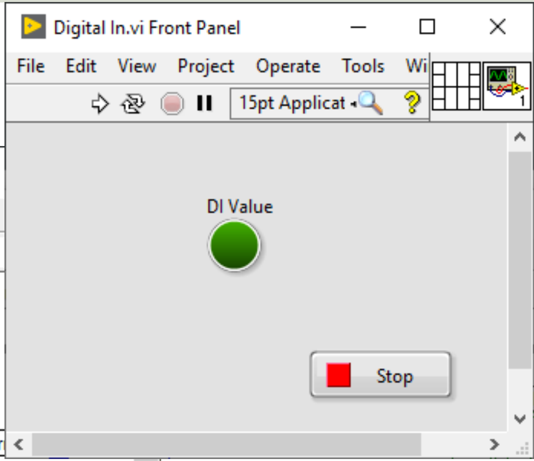


Button is NOT Pushed => False/Low



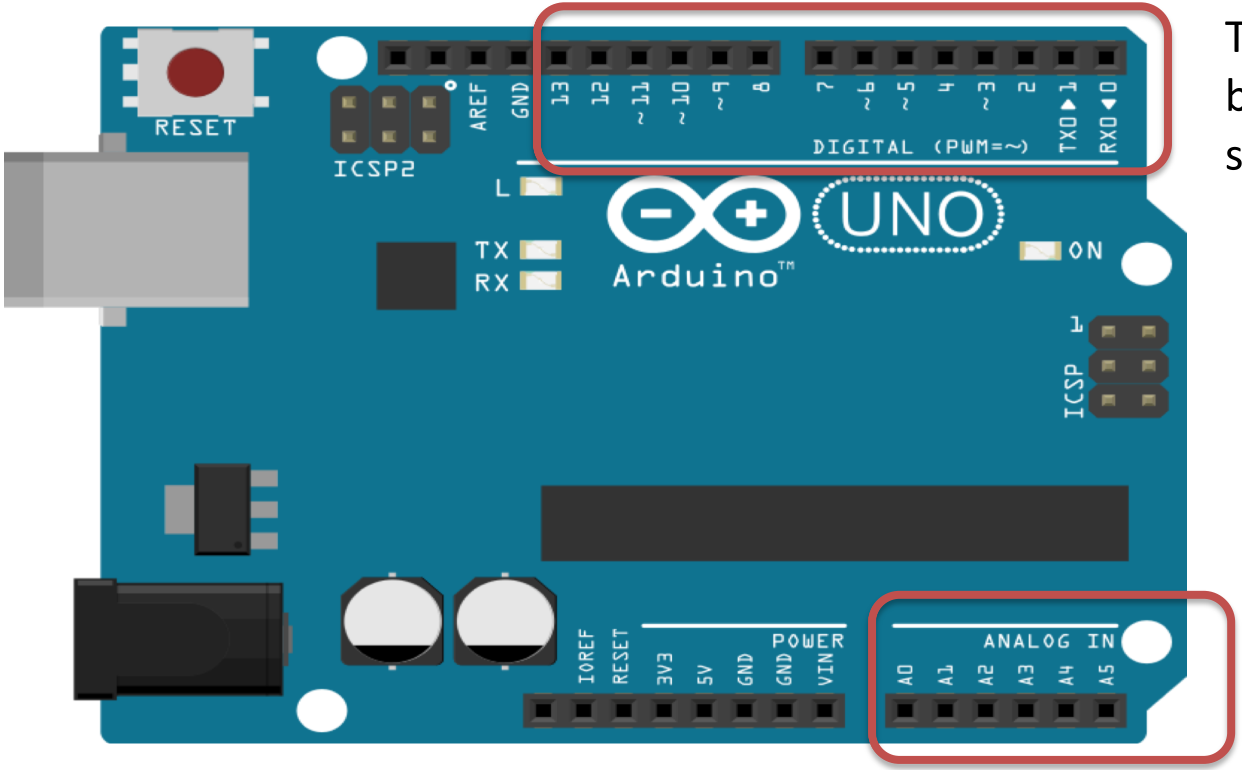
Button is Pushed => True/High





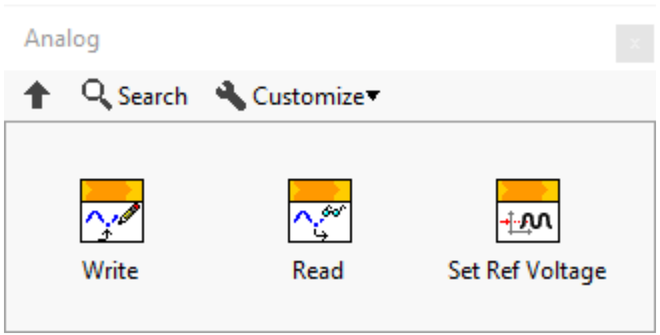
Analog I/O

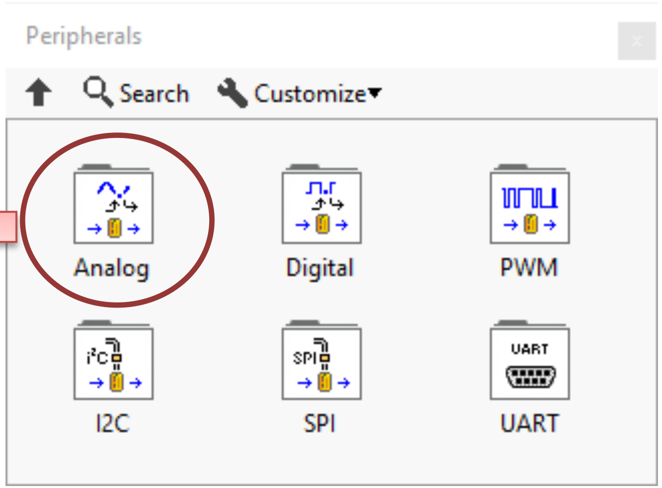
Analog I/O



Those marked with ~ can also be used as "Analog Outputs", so-called PWM outputs

LabVIEW Palette – Analog I/O





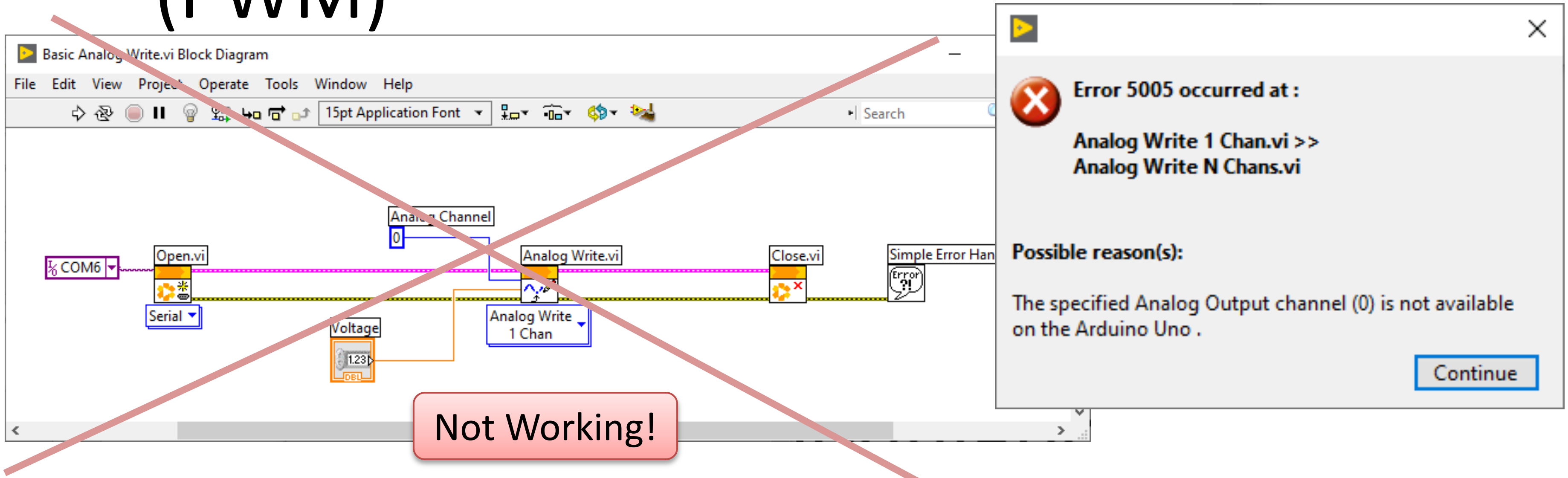
Analog Out (AO)

Hans-Petter Halvorsen

Table of Contents

Analog Out

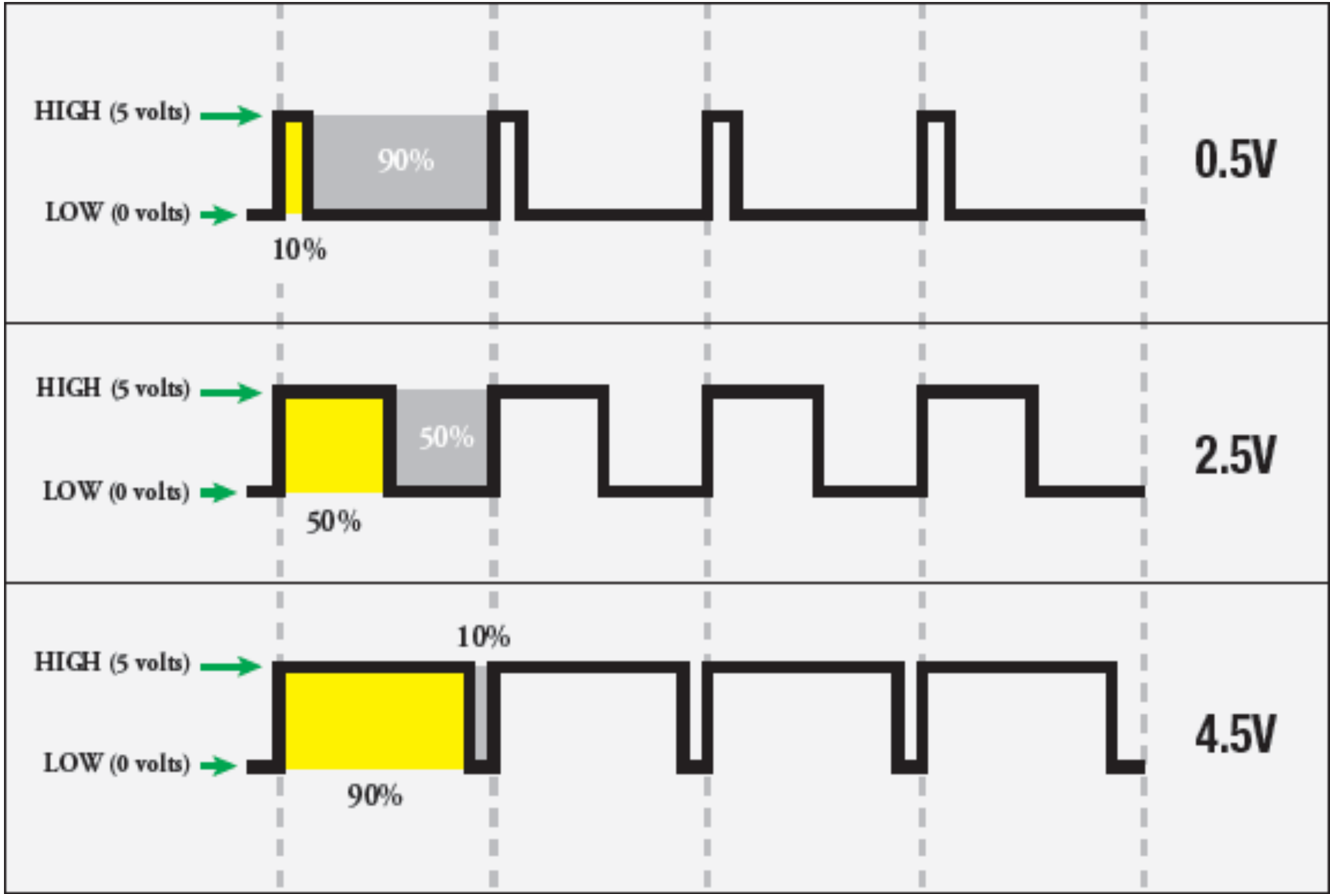
- Arduino UNO has no real Analog Out
- We need to use Pulse Width Modulation (PWM)



PWM as "Analog Out"

The Arduino UNO has no real Analog Out pins, but we can use a PWM pin.

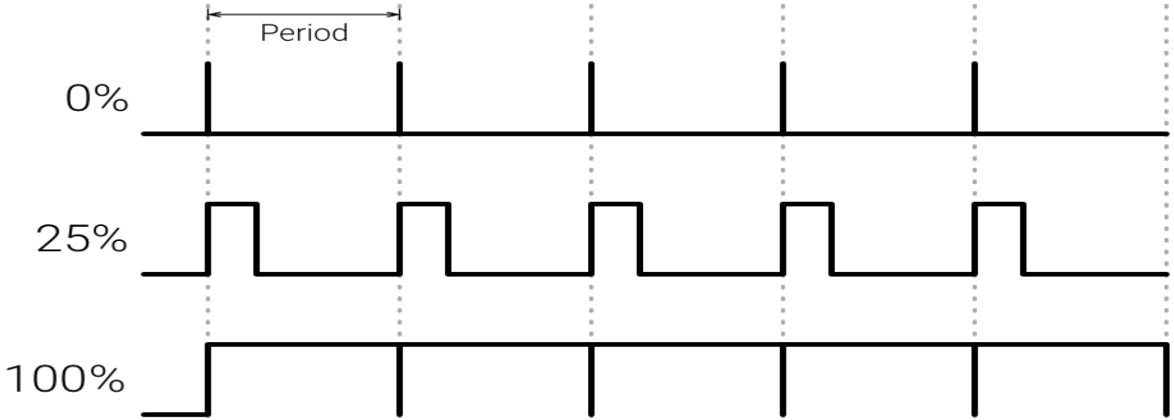
PWM can be used to control brightness of a LED, control the speed of a Fan, control a DC Motor, etc.



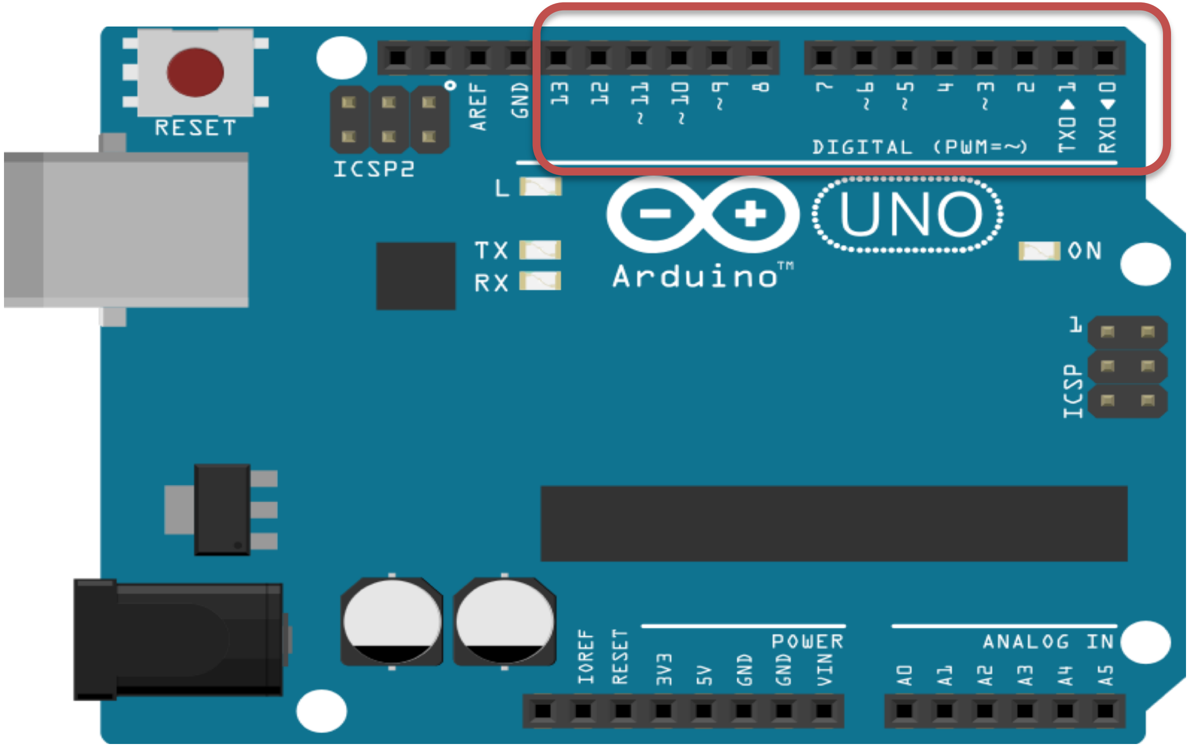
PWM

PWM is a digital (i.e., square wave) signal that oscillates according to a given *frequency* and *duty cycle*.
The frequency (expressed in Hz) describes how often the output pulse repeats.
The period is the time each cycle takes and is the inverse of frequency.
The duty cycle (expressed as a percentage) describes the width of the pulse within that frequency window.

You can adjust the duty cycle to increase or decrease the average "on" time of the signal. The following diagram shows pulse trains at 0%, 25%, and 100% duty:



PWM



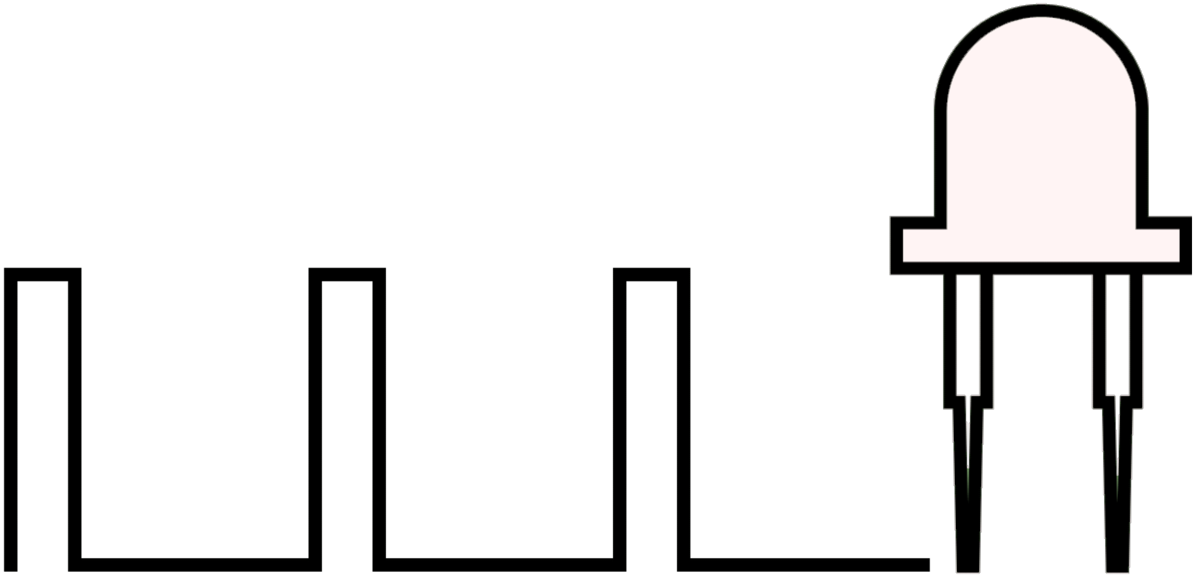
The Digital Pins marked with ~ can be used as "Analog Outputs", so-called PWM outputs

Control Brightness of a LED

- We've seen how to turn an LED on and off, but how do we control its brightness levels?
- An LED's brightness is determined by controlling the amount of current flowing through it, but that requires a lot more hardware components.
- A simple trick we can do is to flash the LED faster than the eye can see!
- By controlling the amount of time the LED is on versus off, we can change its perceived brightness.
- This is known as *Pulse Width Modulation* (PWM).

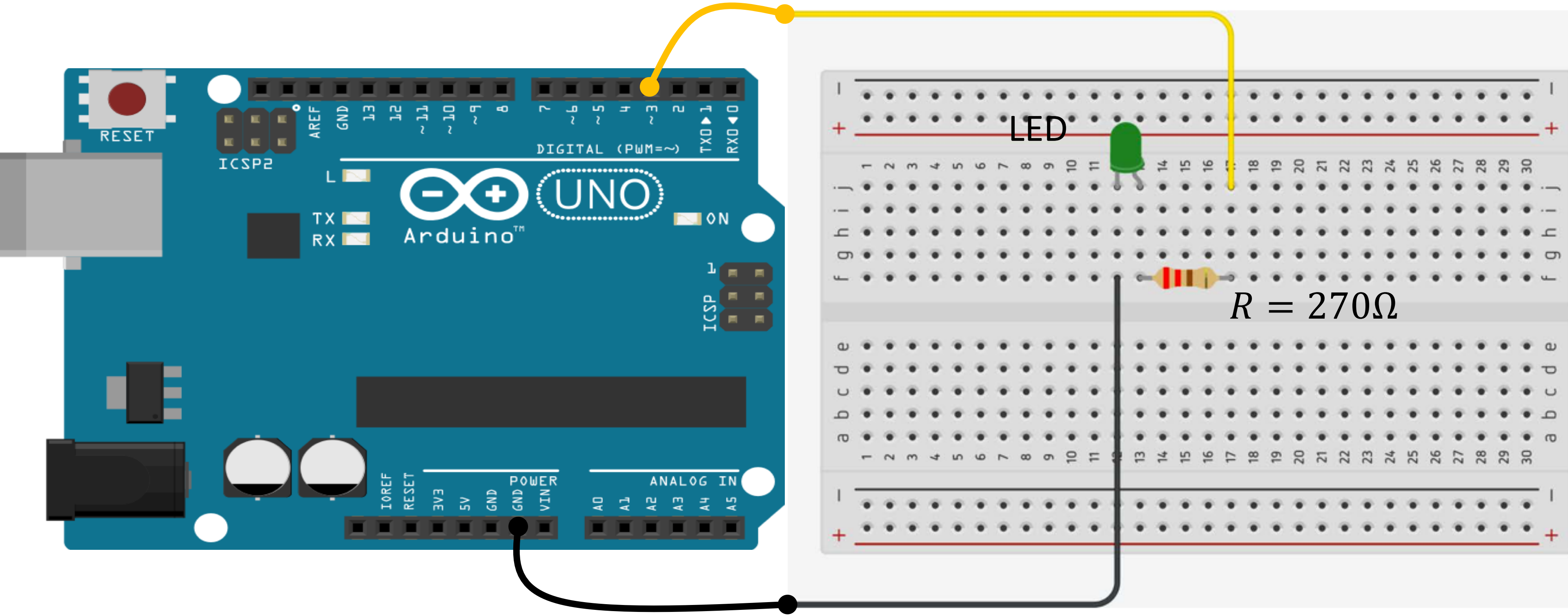
Control Brightness of a LED

Below we see how we can use PWM to control the brightness of a LED



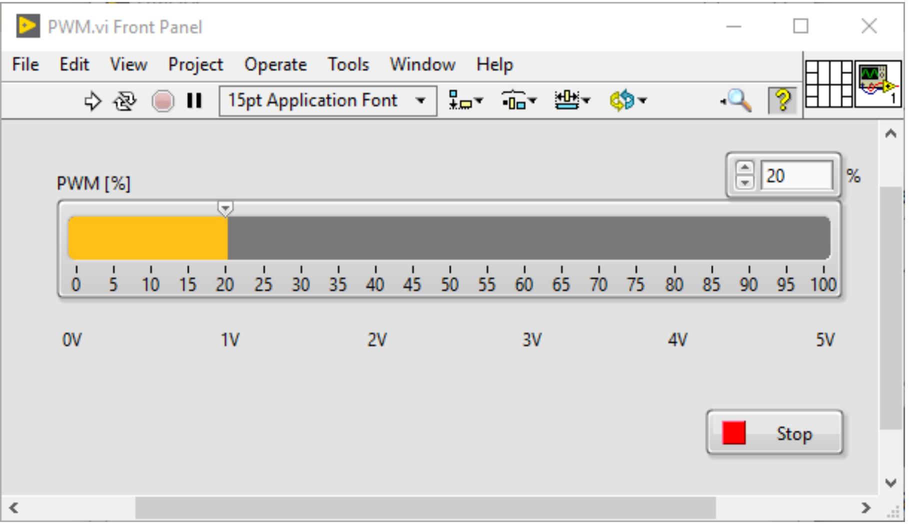
https://www.electronicwings.com/raspberry-pi/raspberry-pi-pwm-generation-using-python-and-c

Wiring



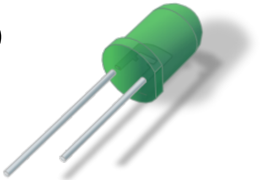
LED
$R = 270\Omega$
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

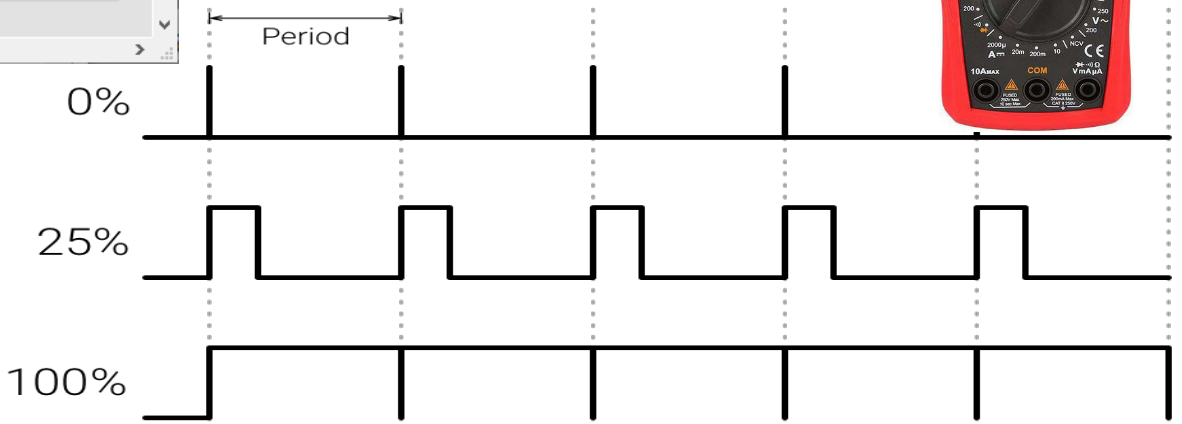
PWM Example



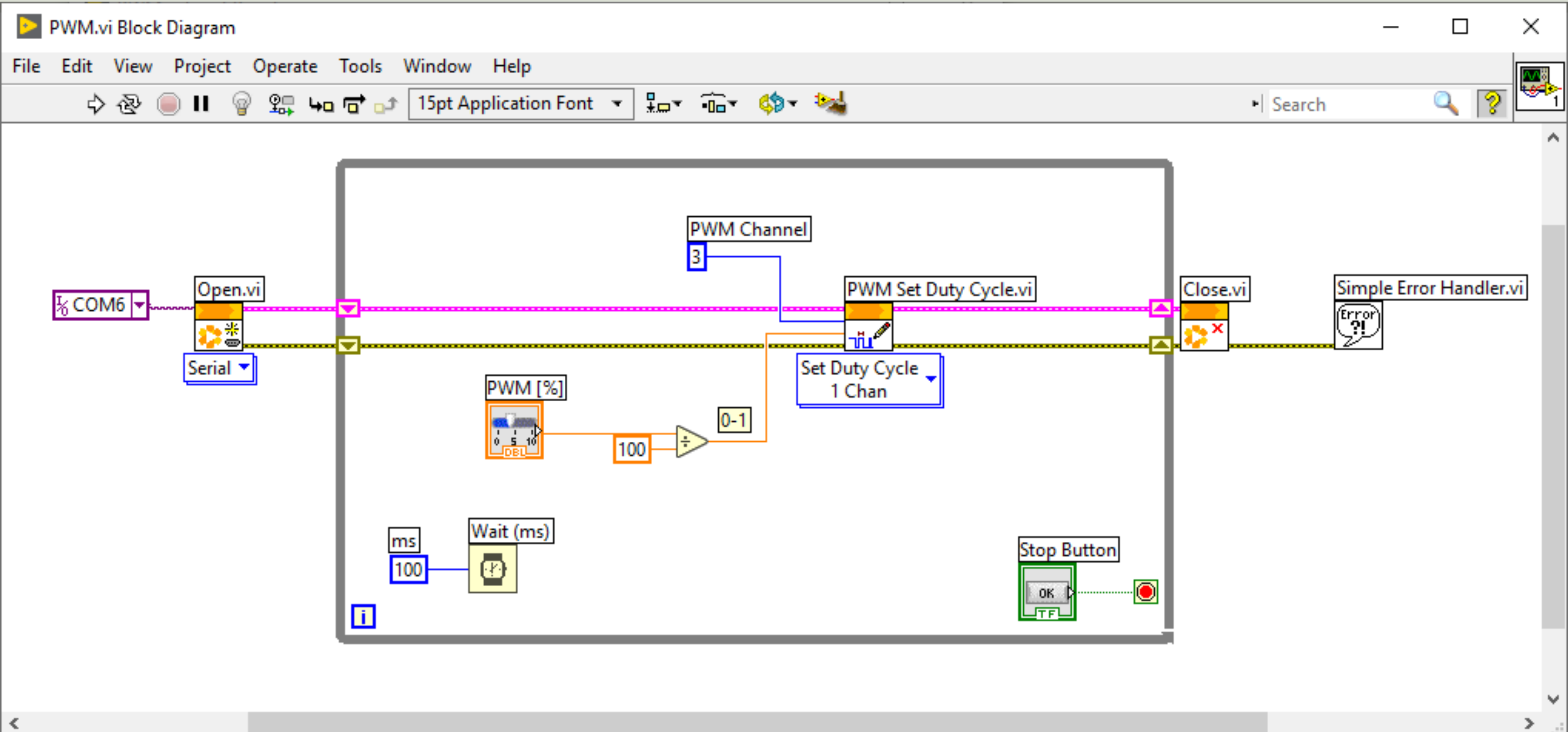
We will see the brightness of the LED will increase.

Or you can use a Multimeter and see the (average) voltage will increase



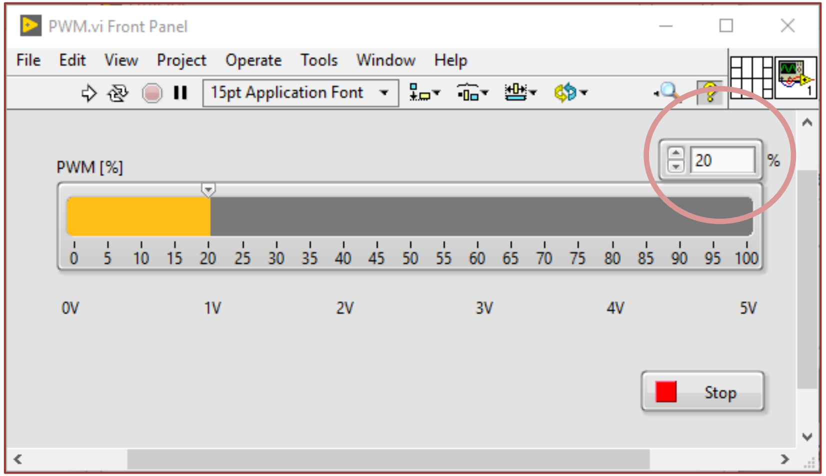


PWM Example



PWM.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
COM6
Open.vi
Serial
PWM Channel
3
PWM Set Duty Cycle.vi
Set Duty Cycle
1 Chan
PWM [%]
100
0-1
Close.vi
Simple Error Handler.vi
Error
ms
100
Wait (ms)
Stop Button
OK

PWM Example



20% → 1V

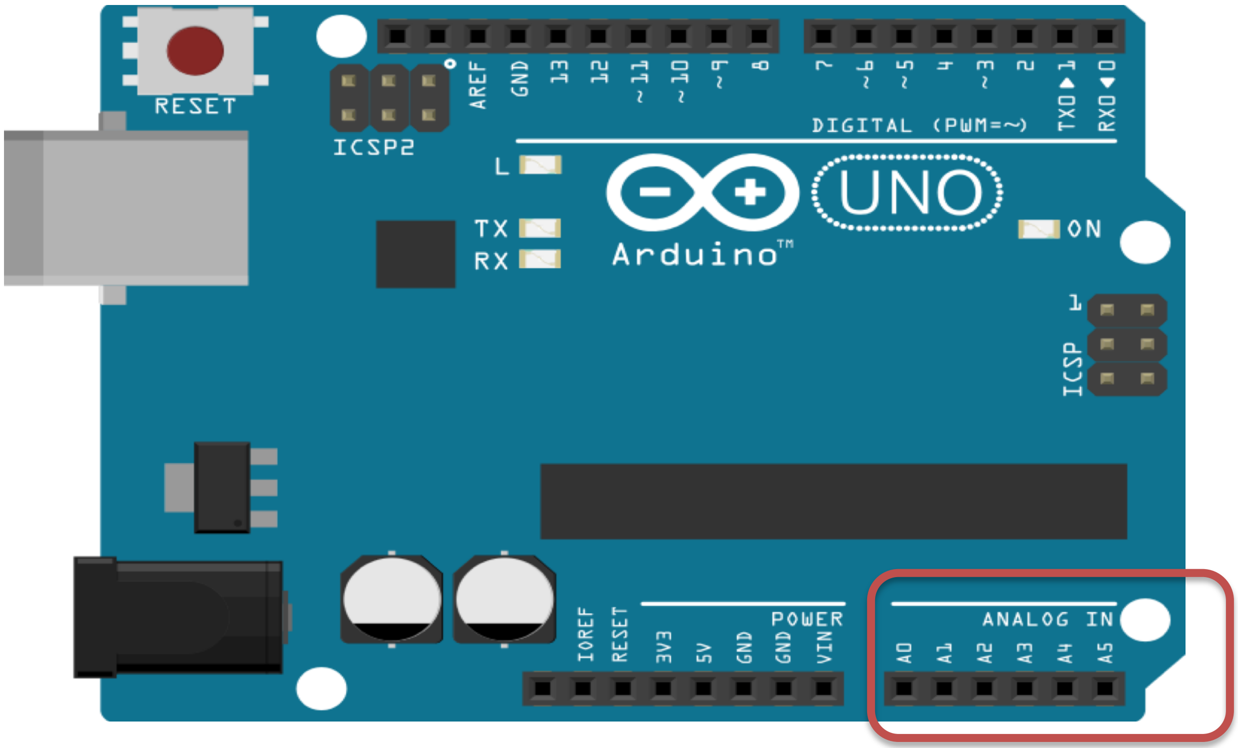
0-100% → 0-5V

Analog In (AI)

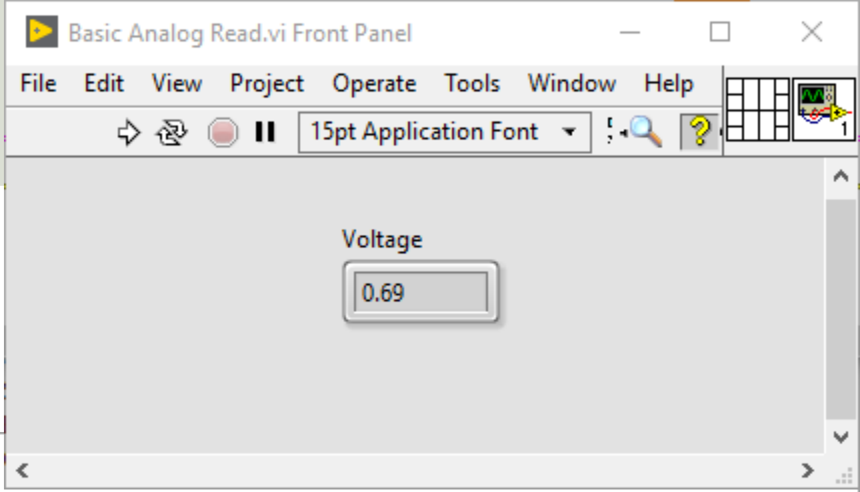
Hans-Petter Halvorsen

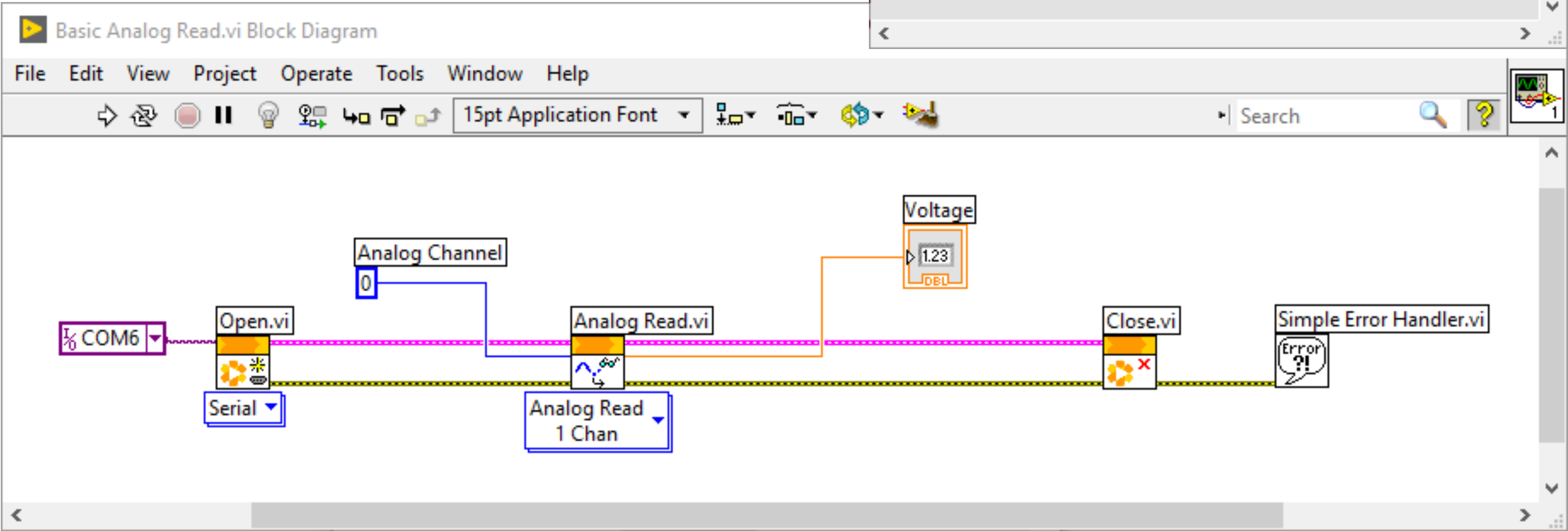
Table of Contents

Analog Input

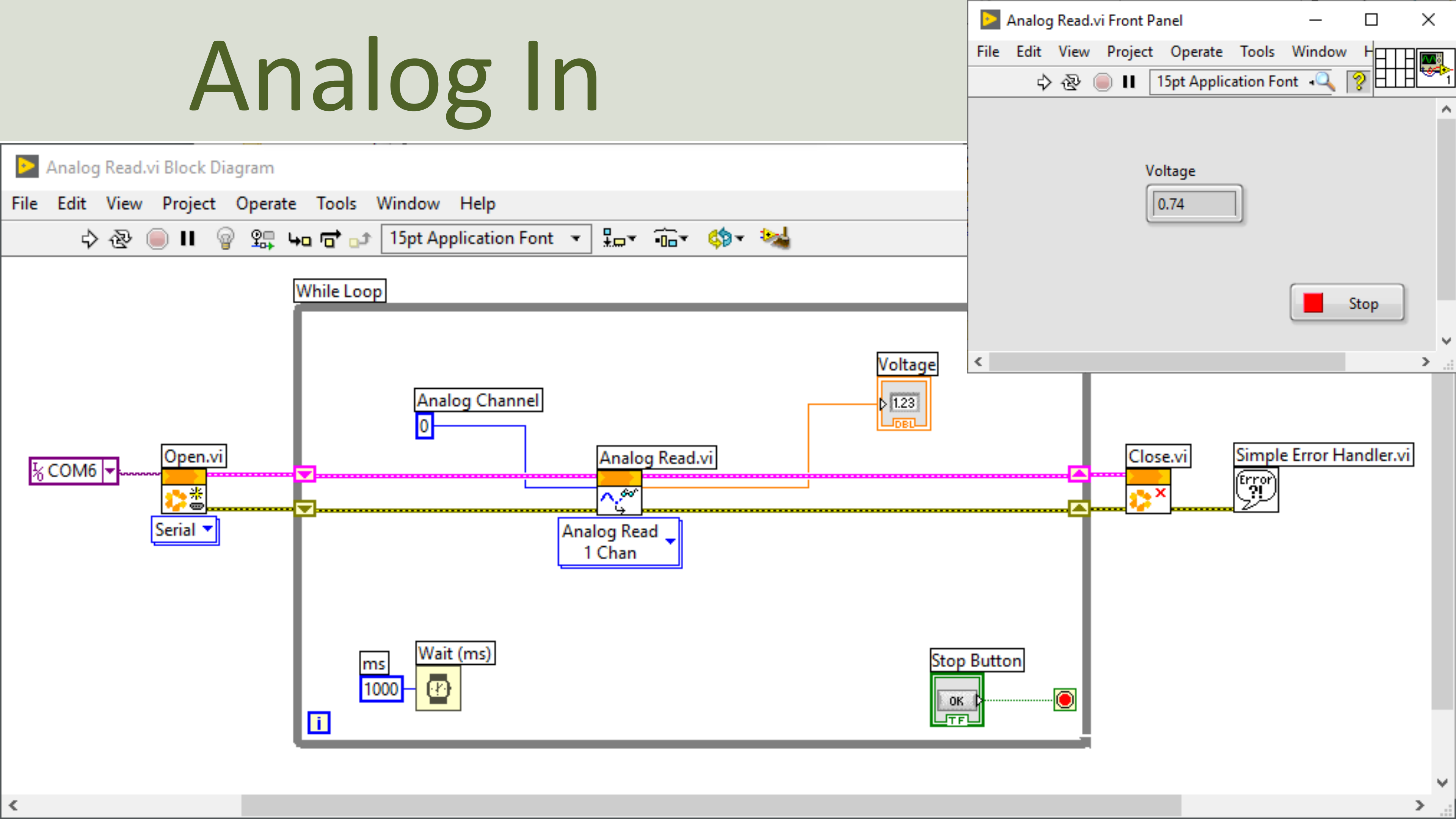


Analog In


Basic Analog Read.vi Front Panel
File Edit View Project Operate Tools Window Help
15pt Application Font
Voltage
0.69


Basic Analog Read.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Voltage
1.23
DBL
Analog Channel
0
COM6
Open.vi
Serial
Analog Read.vi
Analog Read
1 Chan
Close.vi
Simple Error Handler.vi
Error
?!

Analog In



Analog Read.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
While Loop
Voltage
1.23
DBL
Analog Channel
0
COM6
Open.vi
Serial
Analog Read.vi
Analog Read
1 Chan
Close.vi
Simple Error Handler.vi
Error
?!
ms
1000
Wait (ms)
i
Stop Button
OK
TF
Analog Read.vi Front Panel
File Edit View Project Operate Tools Window
15pt Application Font
Voltage
0.74
Stop

LabVIEW LINX Arduino Temperature Sensors

Hans-Petter Halvorsen

Table of Contents

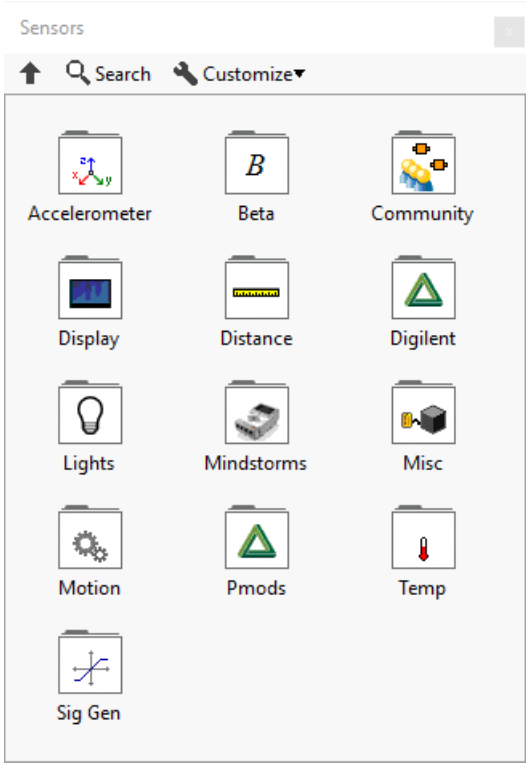
Sensors

Sensors

We will use 2 different types of Temperature Sensors:

- TMP36 Temperature Sensor
- Thermistor Temperature Sensor

Sensors



Some Examples of premade Sensor VIs

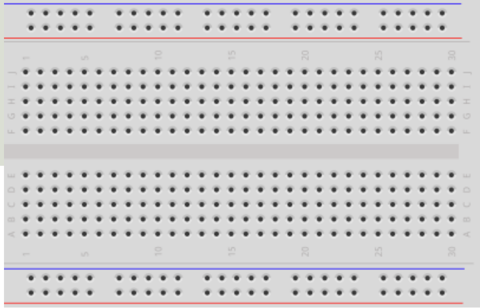
You can use these, or you can also easily make your own VIs for interfacing Sensors from scratch

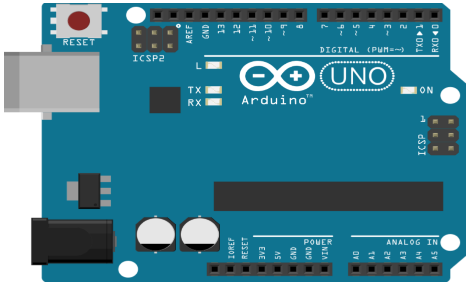
TMP36 Temperature Sensor

Hans-Petter Halvorsen

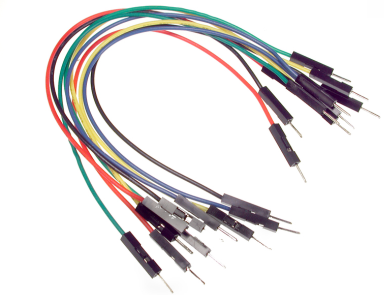
Table of Contents

Hardware

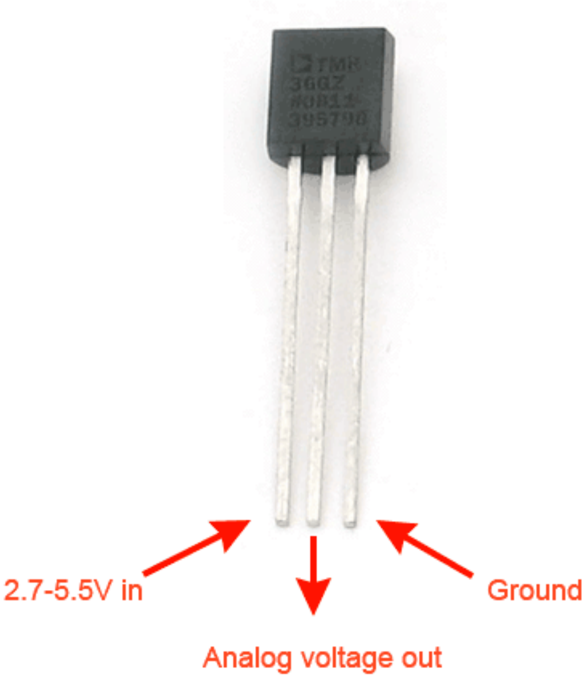




- Arduino
- Breadboard
- TMP36 Temperature Sensor
- Wires (Jumper Wires)



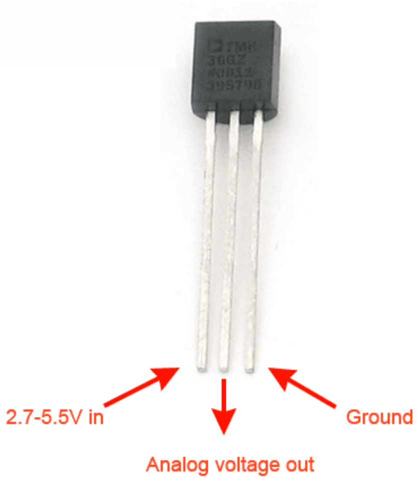
TMP36 Temperature Sensor



- A Temperature sensor like TM36 use a solid-state technique to determine the temperature.
- They use the fact as temperature increases, the voltage across a diode increases at a known rate.
- It costs only about $1

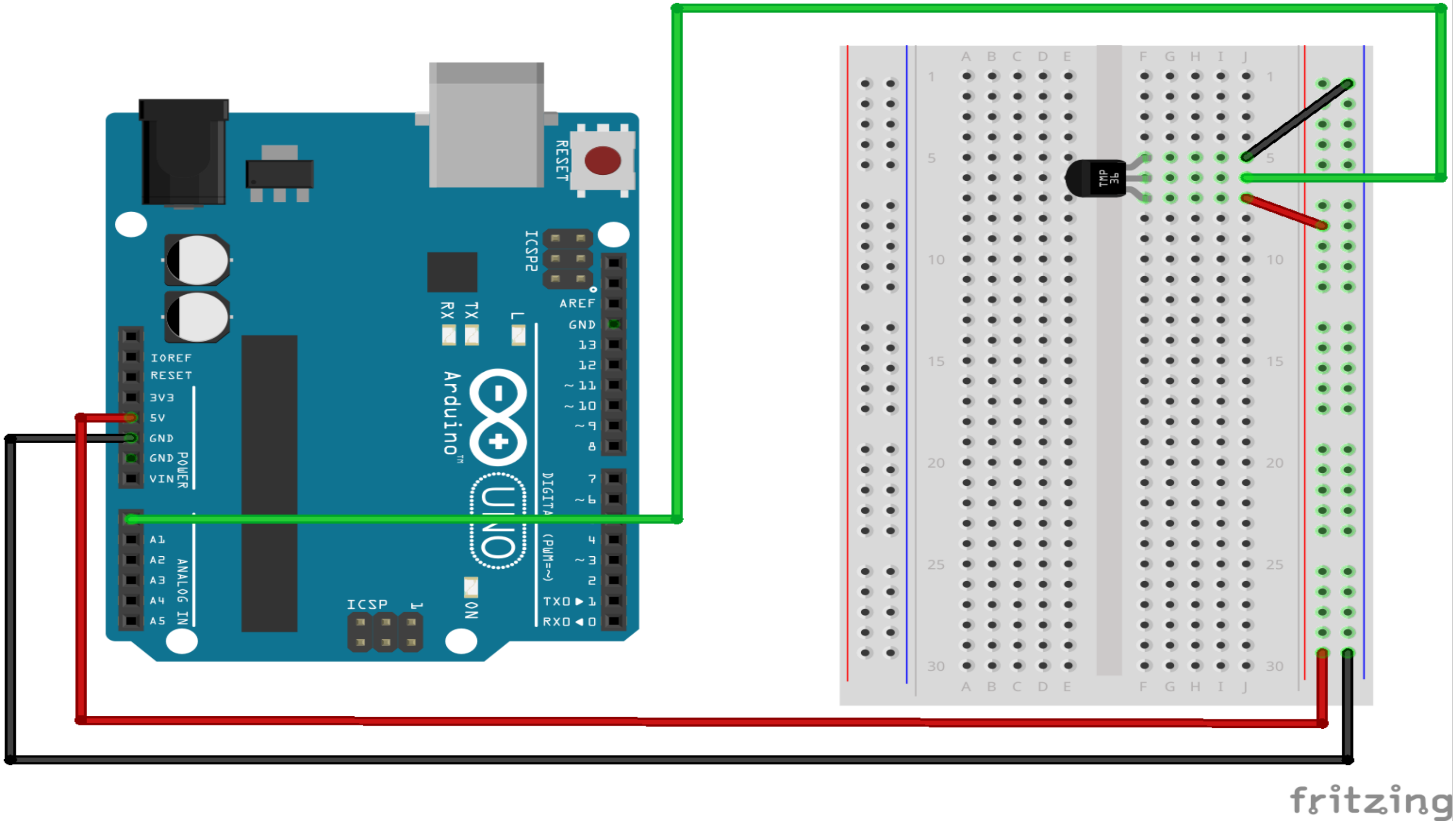
https://learn.adafruit.com/tmp36-temperature-sensor

TMP36



TMP is a small, low-cost temperature sensor and cost about $1 (you can buy it “everywhere”)

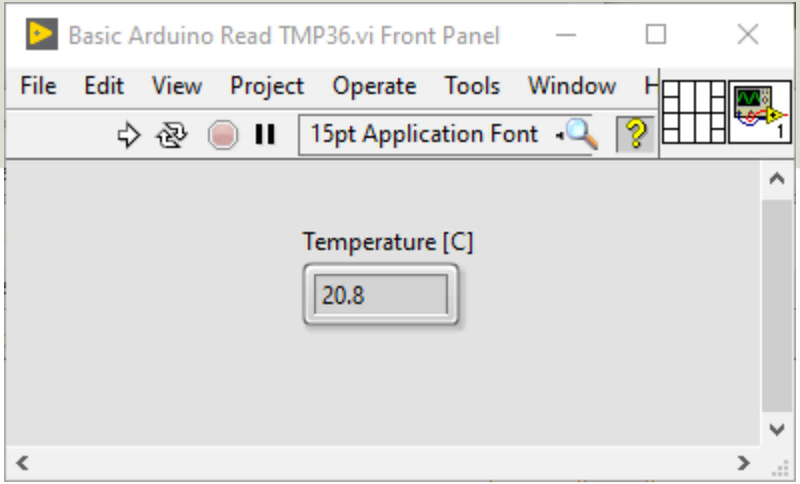
Wiring

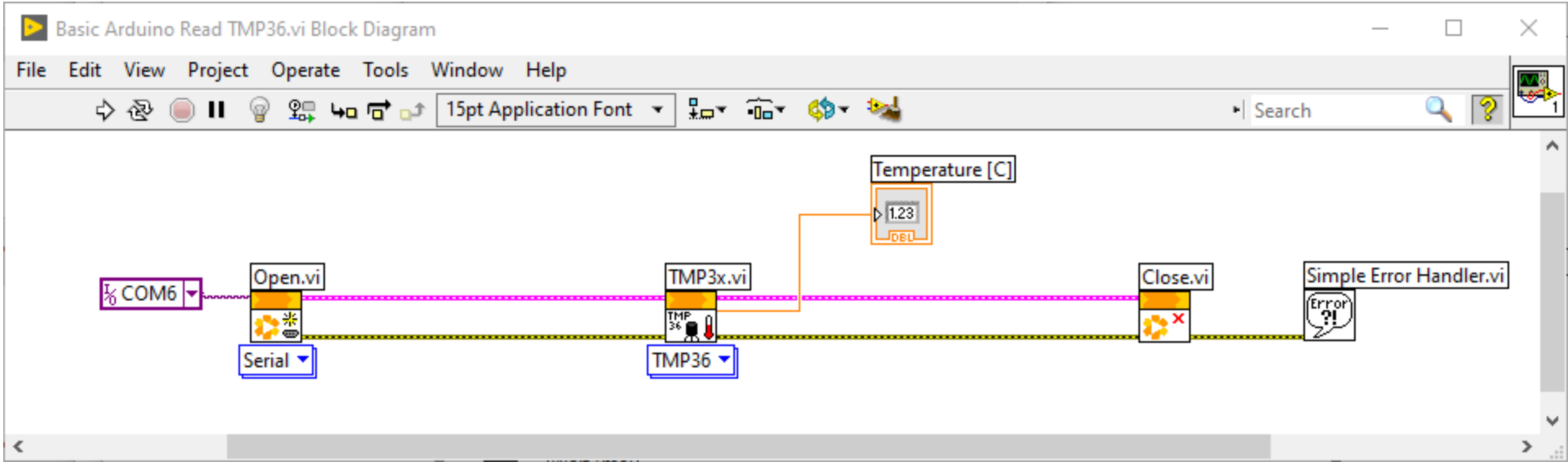


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
TX
RX
L
Arduino™
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A1
A2
A3
A4
A5
ANALOG IN
TMP 36
A B C D E
F G H I J
1
5
10
15
20
25
30
fritzing

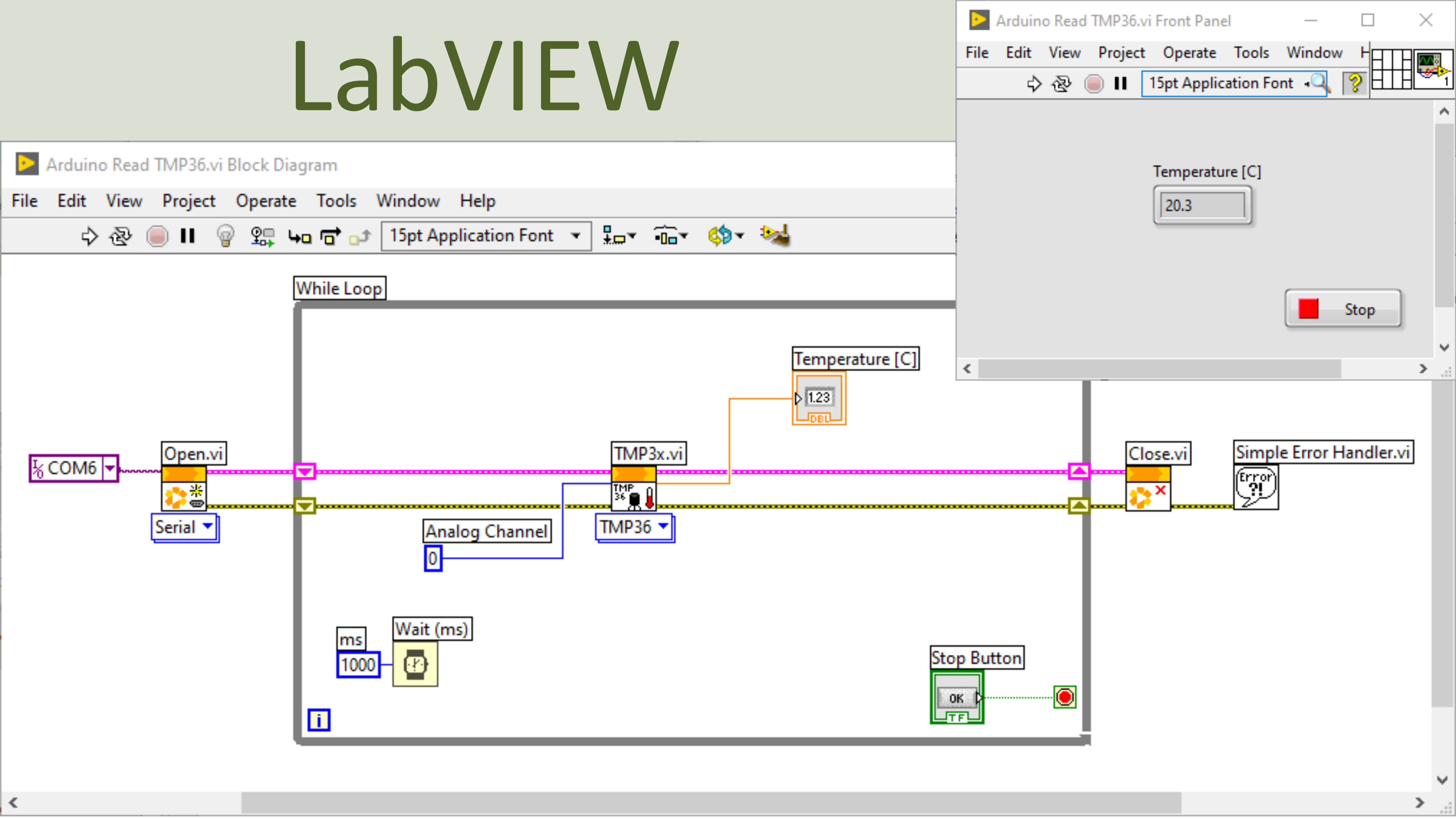
LabVIEW

LabVIEW LINX has a built-in SubVI for TMP3x Sensors





LabVIEW



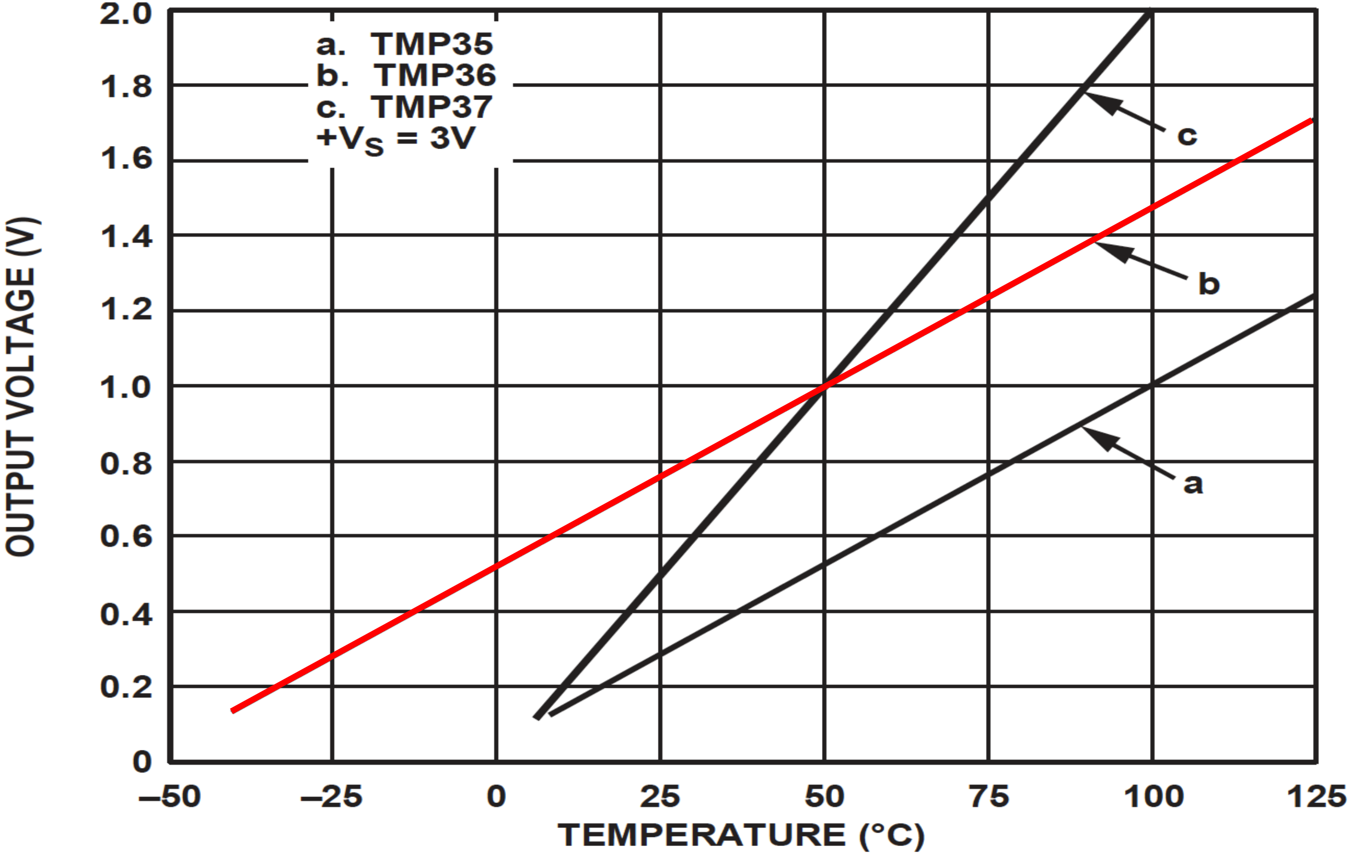
Arduino Read TMP36.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
While Loop
COM6
Open.vi
Serial
Temperature [C]
1.23
DBL
TMP3x.vi
TMP36
Analog Channel
0
ms
1000
Wait (ms)
Stop Button
OK
TF
Close.vi
Simple Error Handler.vi
Error ?!
Arduino Read TMP36.vi Front Panel
Temperature [C]
20.3
Stop

Do it from Scratch

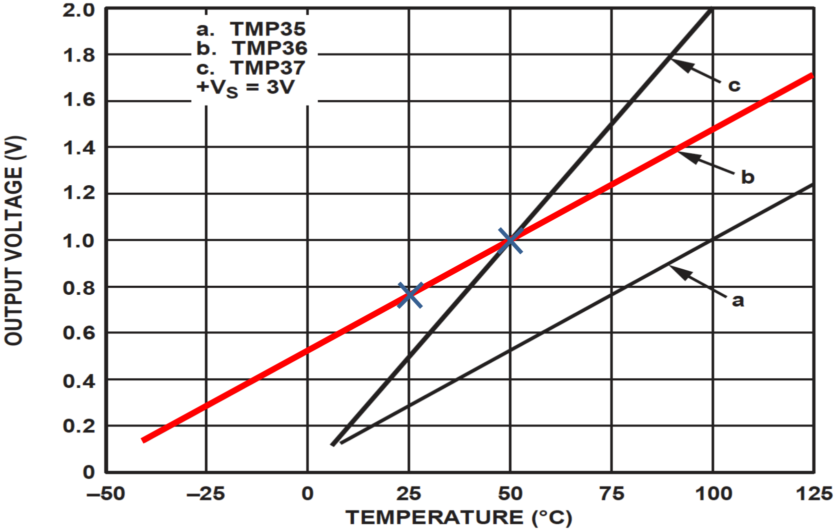
- LabVIEW LINX has a built-in SubVI for TMP3x Sensors (TMP3x.vi)
- Let's see how we can do it from "Scratch"
- We use the Datasheet and the ordinary "Analog Read.vi"

Datasheet

Output Voltage vs. Temperature



Linear Scaling



Convert form Voltage (V) to degrees Celsius
From the Datasheet we have:

$$(x_1, y_1) = (0.75V, 25°C)$$
$$(x_2, y_2) = (1V, 50°C)$$

There is a linear relationship between Voltage and degrees Celsius:

$$y = ax + b$$

We can find a and b using the following known formula:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

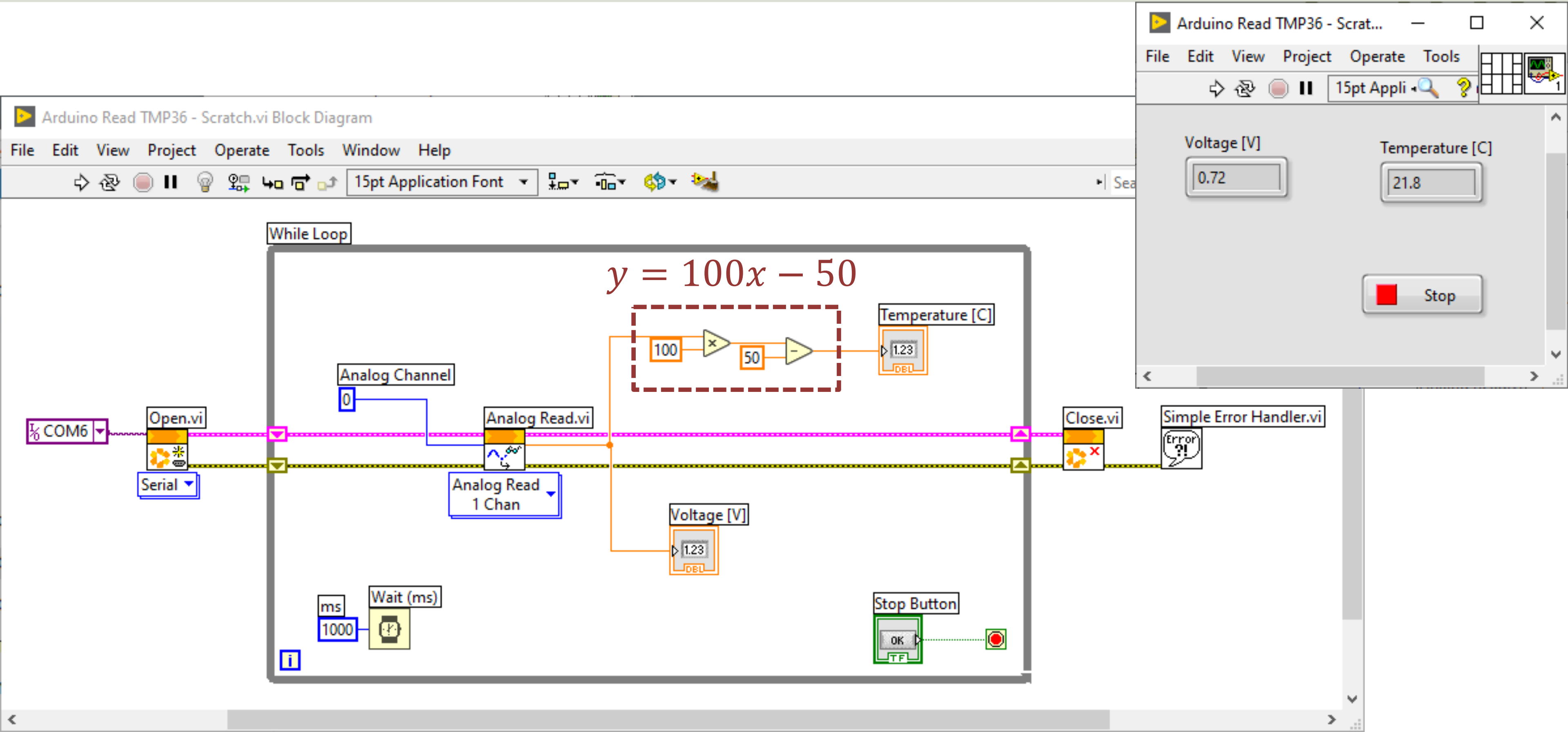
This gives:

$$y - 25 = \frac{50 - 25}{1 - 0.75}(x - 0.75)$$

Then we get the following formula:

$$y = 100x - 50$$

LabVIEW Example



https://www.halvorsen.blog

Thermistor Temperature Sensor

Hans-Petter Halvorsen

Table of Contents

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 a 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**):

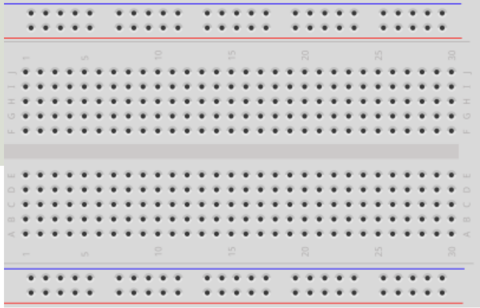
[Wikipedia]

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

where A, B, C are constants given below

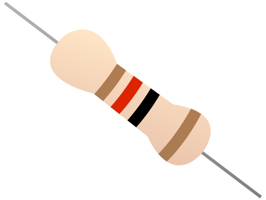
$A = 0.001129148, B = 0.000234125 \; and \; C = 8.76741E - 08$

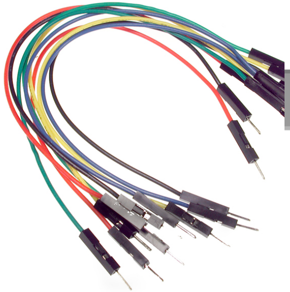
Hardware

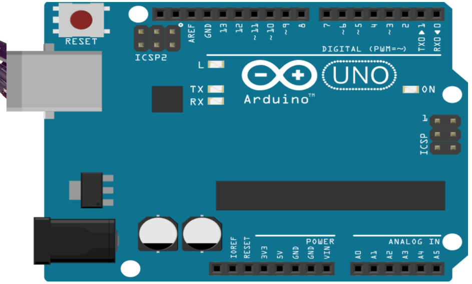


- Arduino
- Breadboard

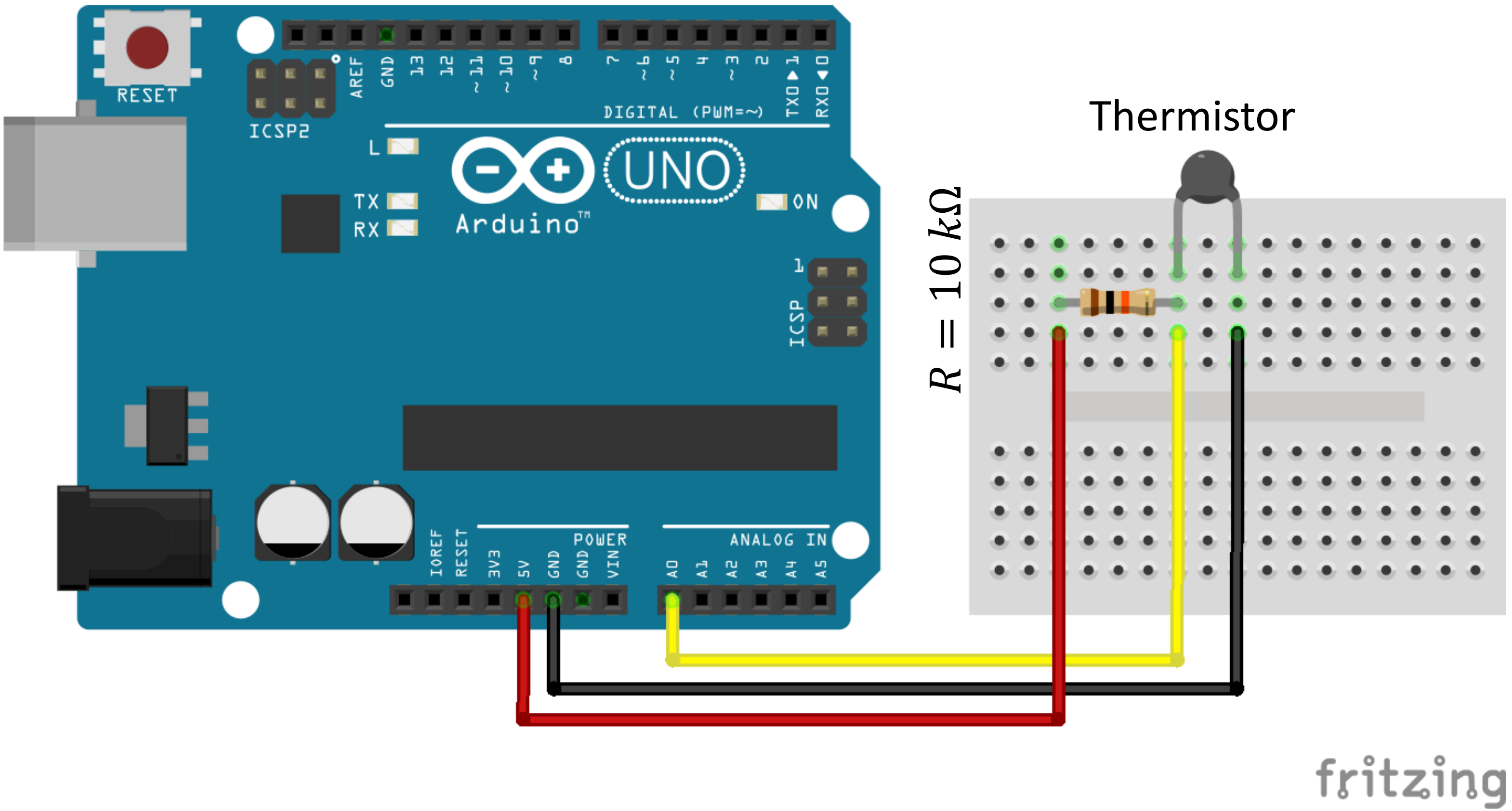
- Thermistor 10K (Temperature Sensor)
- Wires (Jumper Wires)
- Resistor 10 kΩ







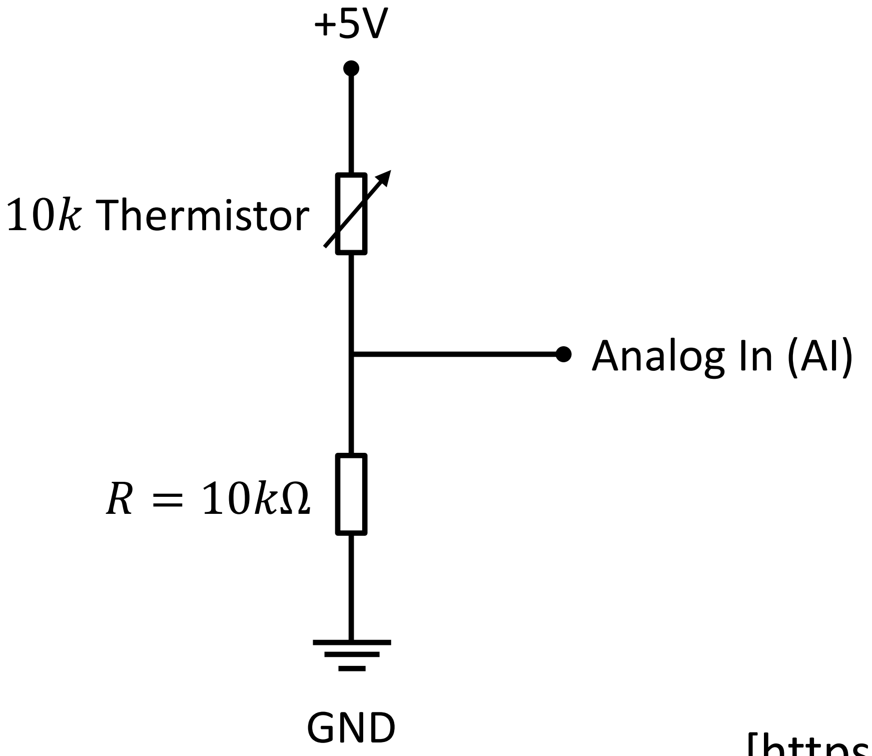
Wiring



Thermistor
$R = 10\ k\Omega$
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX0▸1
RX0◂0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
1
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
fritzing

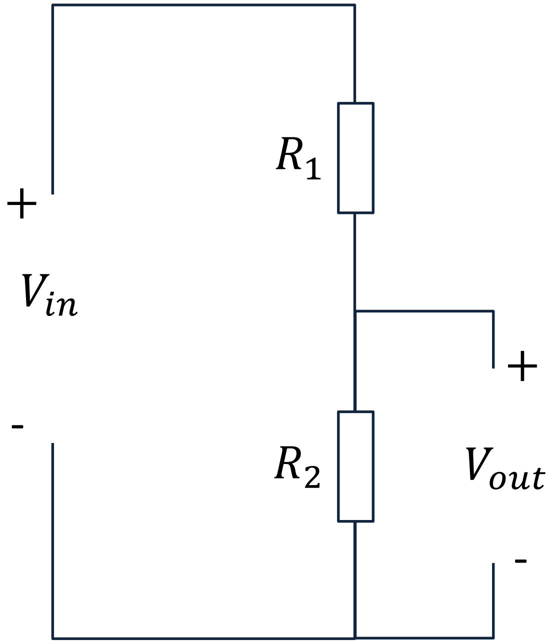
Voltage Divider

The wiring is called a “Voltage divider”:



[https://en.wikipedia.org/wiki/Voltage_divider]

General Voltage Divider



We want to find V_{out}

Formula:

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

https://learn.sparkfun.com/tutorials/voltage-dividers/all

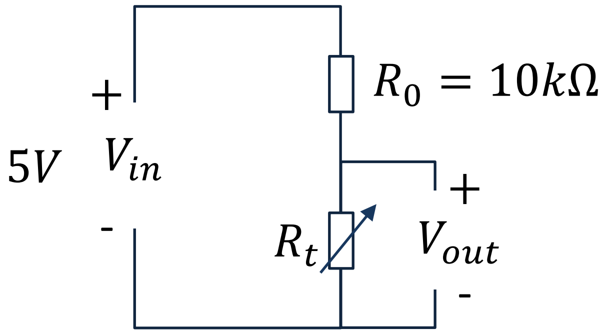
Voltage Divider for our System

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℃

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

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:

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

Where the Temperature T_K is in Kelvin

$A, B\ and\ C$ are constants

$$A = 0.001129148$$
$$B = 0.000234125$$
$$C = 0.0000000876741$$

The Temperature in degrees Celsius will then be:

$$T_C = T_K - 273.15$$

Pseudo Code

1. Get V_{out} from the DAQ device (Arduino UNO)

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

Pseudo Code

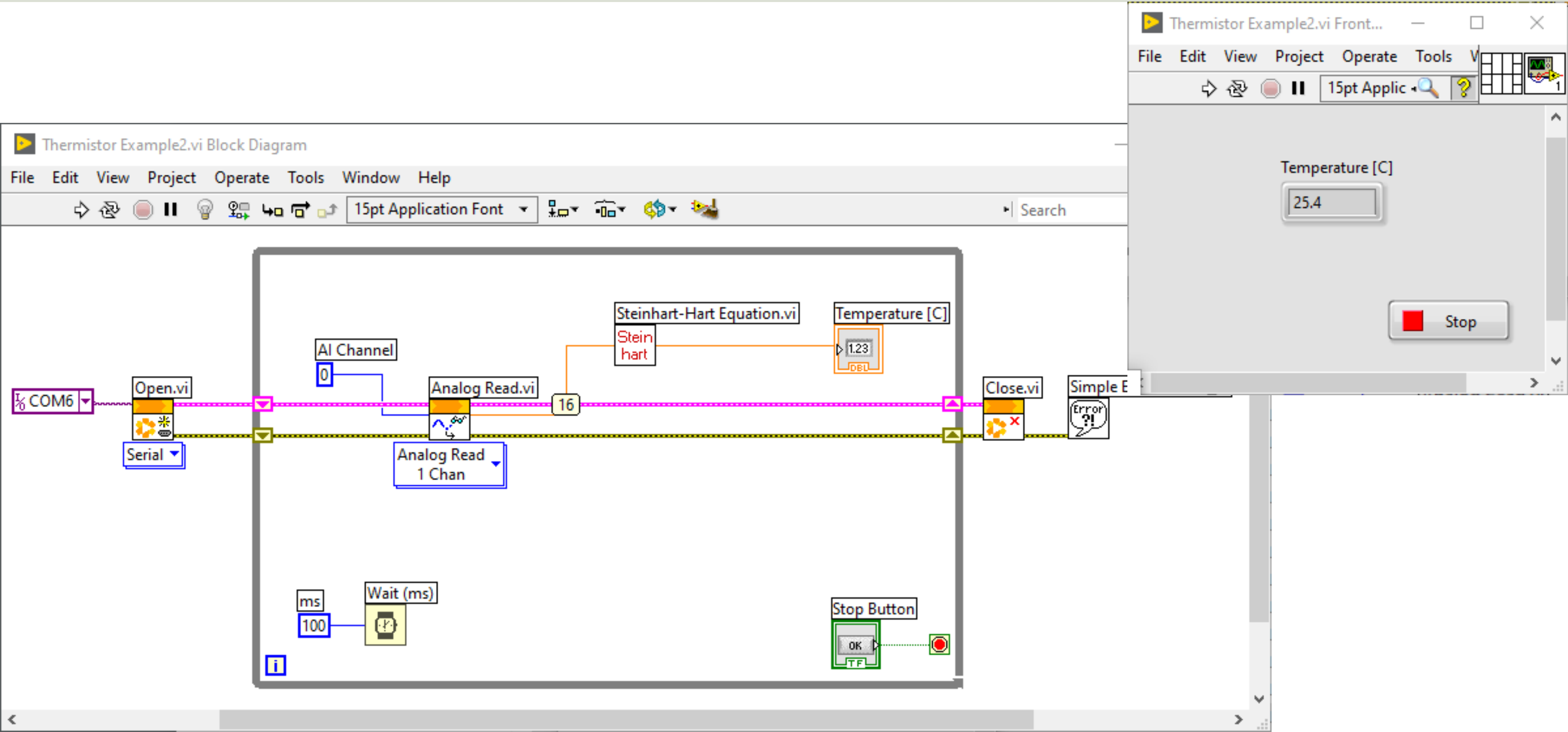
```
float Vin = 5;
float Ro=10000;
float Rt = (Vout*Ro)/(Vin-Vout);

//Steinhart constants
float A = 0.001129148;
float B = 0.000234125;
float C = 0.0000000876741;

//Steinhart-Hart Equation
float TempK = 1 / (A + (B * ln(Rt)) + (C * ln(Rt)**3));

//Convert from Kelvin to Celsius
float TempC = TempK - 273.15;
```

LabVIEW

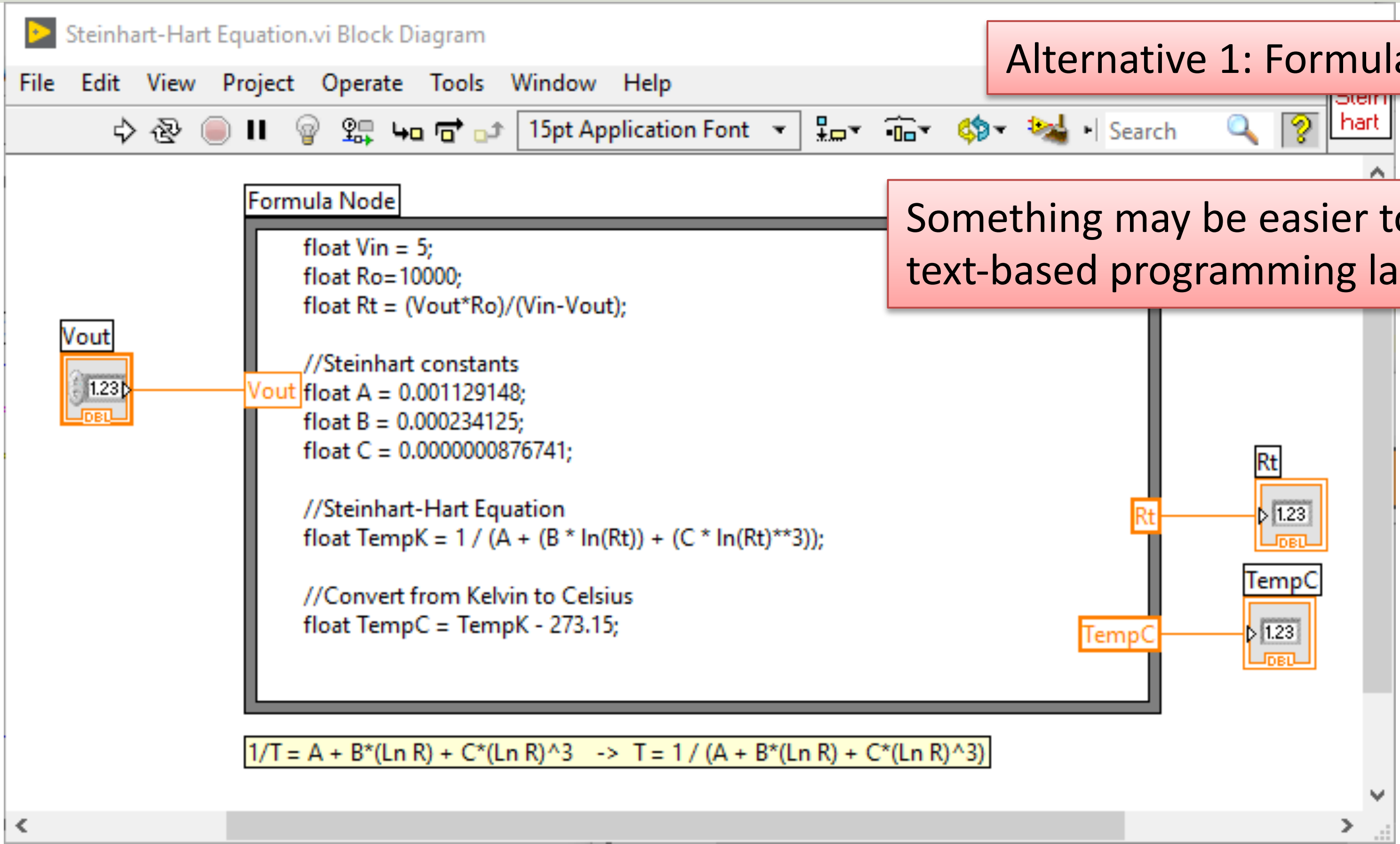


Thermistor Example2.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
COM6
Open.vi
Serial
AI Channel
0
Analog Read.vi
Analog Read
1 Chan
16
Steinhart-Hart Equation.vi
Stein
hart
Temperature [C]
1.23
DBL
Close.vi
Simple E
ms
100
Wait (ms)
Stop Button
OK
TF
Thermistor Example2.vi Front...
File Edit View Project Operate Tools
15pt Applic
Temperature [C]
25.4
Stop

LabVIEW - Steinhart-Hart Equation

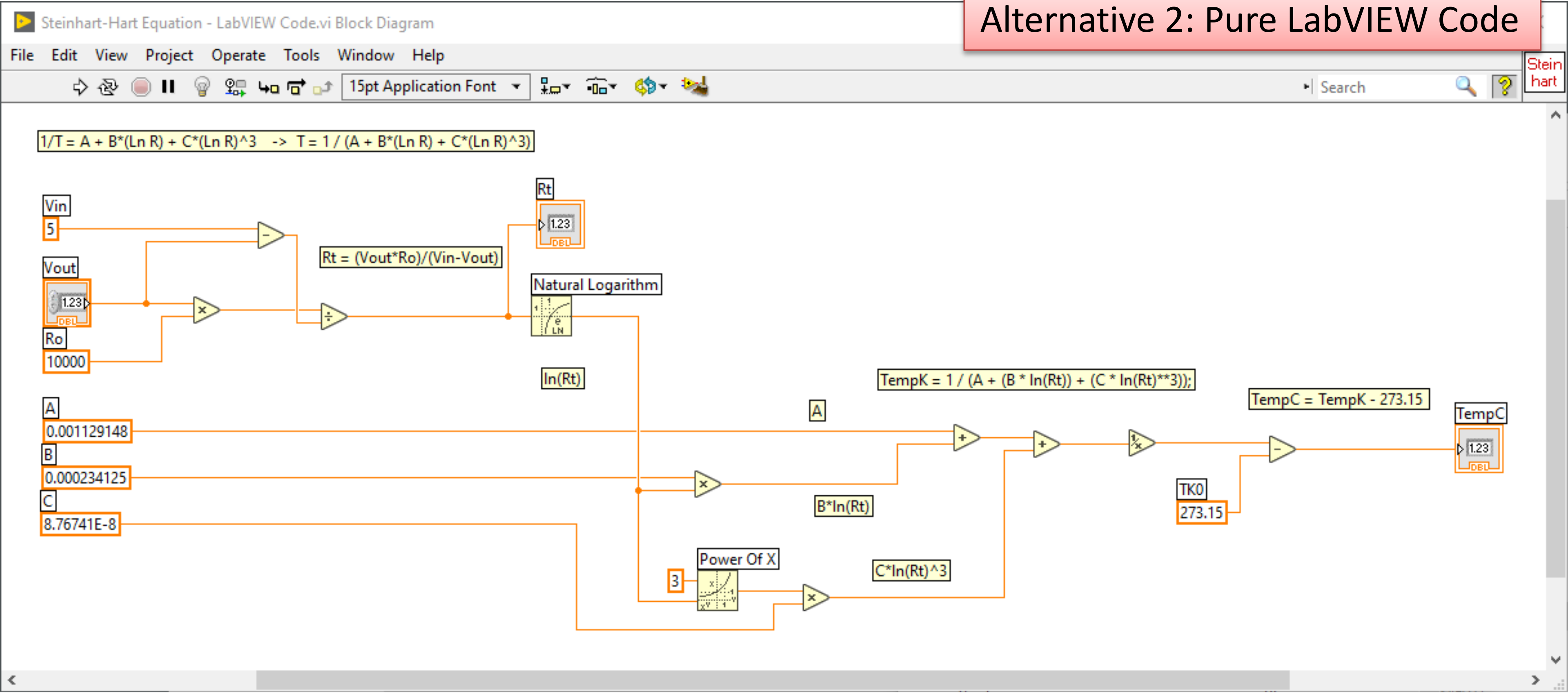
Alternative 1: Formula Node

Something may be easier to do in a text-based programming language



LabVIEW - Steinhart-Hart Equation

Alternative 2: Pure LabVIEW Code



Summary

- This Tutorial has shown how we can use Arduino in combination with the LabVIEW Programming environment
- “LabVIEW LINX Toolkit” is an add-on for LabVIEW which makes it possible to program the Arduino device using LabVIEW
- In that way we can create Data Logging Applications, etc. without the need of an expensive DAQ device
- If you in addition use the “LabVIEW Community Edition” (free for non-commercial use) you get a very low-cost DAQ/Datalogging System!
- You can also easily add features for logging data to Files or a Database System like SQL Server, or an OPC Server

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog