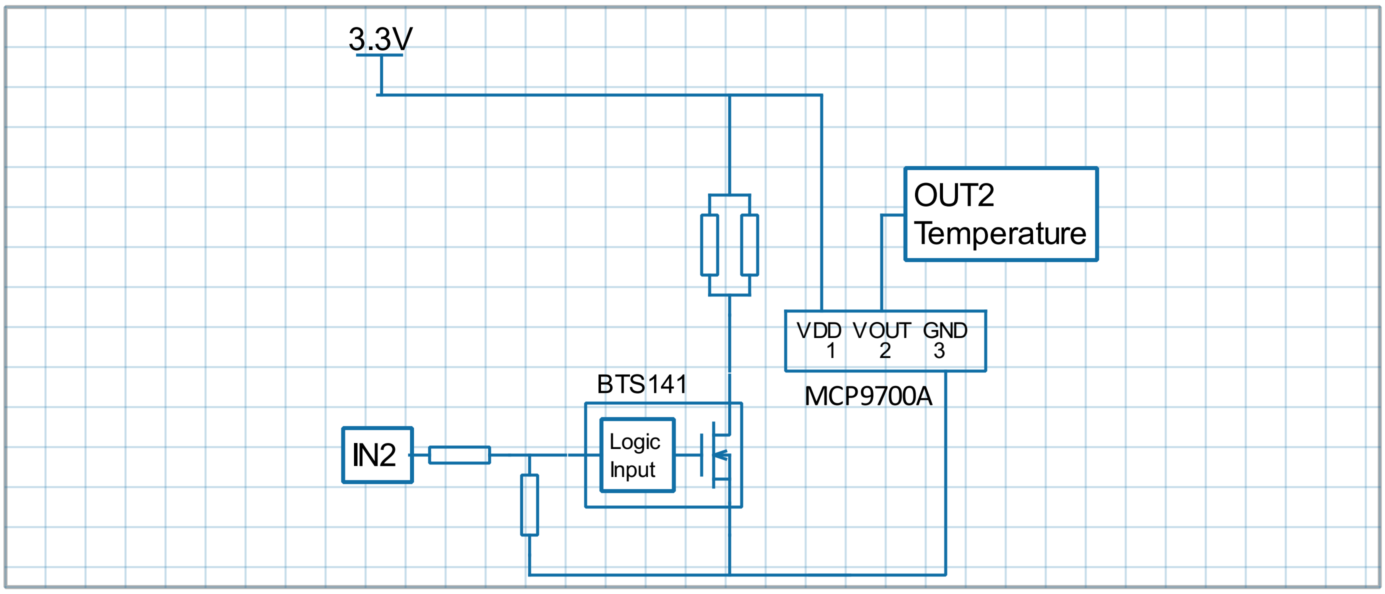
Temperature Control using LabJack and MatDeck PID controller, with GUI Configuration

This document illustrates how the LabJack T7 device can be used to control ambient temperatures by switching the electronic circuit on and off. In the case when the circuit is switched on, the current through the resistors produces heating. Temperature measurement is performed by using MCP9701A temperature sensors. LabJack T7 is used to switch on/off the circuit using a digital output (DIO) and to measure the temperature using an analog input AIN. The control signal, PWM signal duty cycle, is set using PID controller which is implemented in MatDeck. MatDeck provides GUI forms for effective and intuitive configuration of LabJack devices, as illustrated here.

Schematics of the electronic circuits

The schematics of the described system for temperature control is displayed below. It should be pointed out that the schematics is created in MatDeck, which is suitable for various professional drawings.



The description of circuit is as follows:
Functionality

- IN2 and OUT2 demonstrate PID temperature control

Demo board schematic pin descriptions

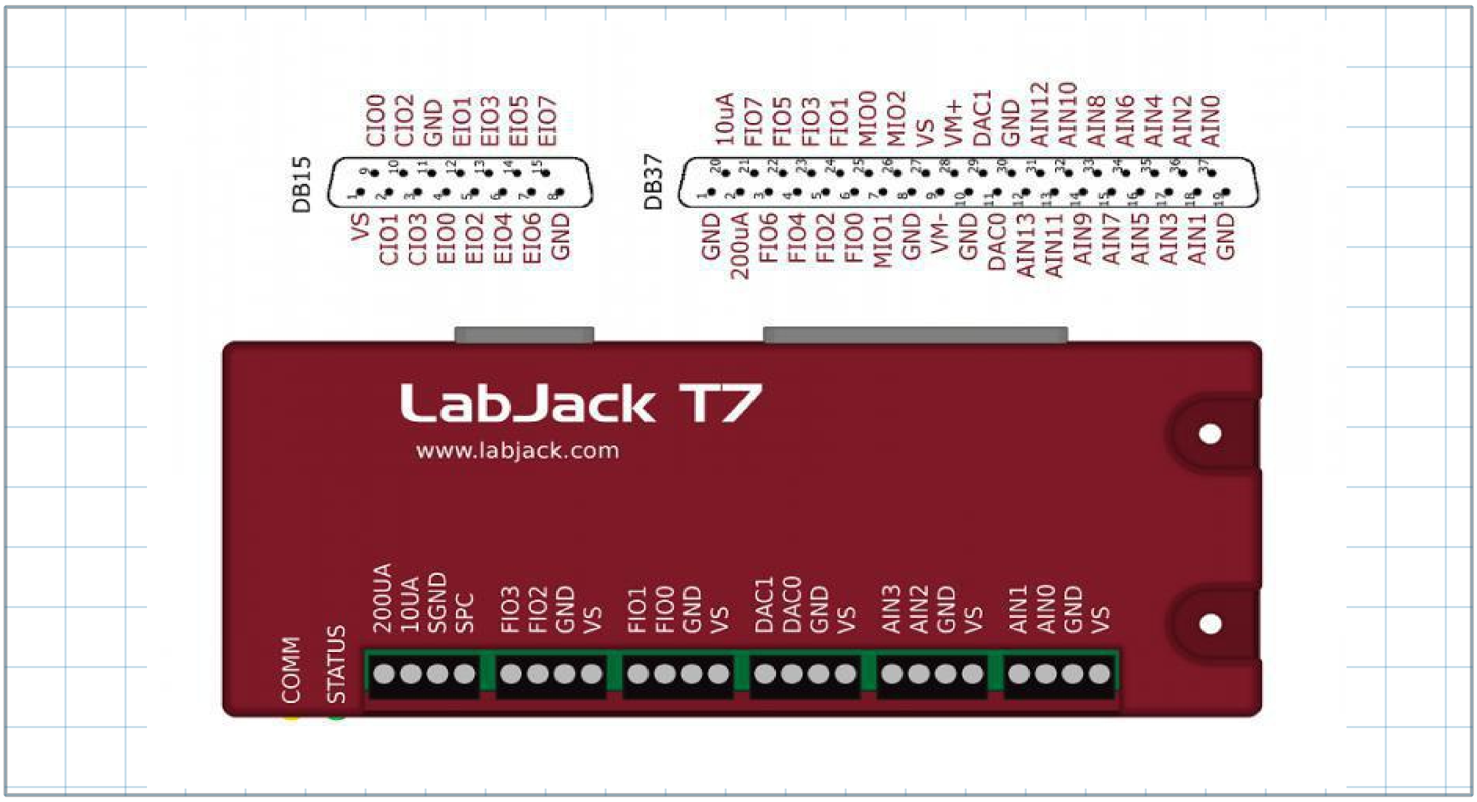
- IN2 -input for PWM driver to heat up the resistor
- OUT2 - output from the temperature sensor in mV

Connection to ADC unit

- IN2 is connected to the PWM output.
- OUT2 is connected to Analog inputs 2.

Parts:

- For power driver is used BTS 141 which is logic level low side driver.
- Temperature sensor MCP9700A-E/TO



Use of LabJack T7

In this experiment LabJack T7 is used to produce a digital PWM output which is connected to the IN2 signal. At the same time, T7 is used to measure the temperature by collecting OUT2 signals at the AIN2 channel. MatDeck supports LabJack functions which can be used directly inside MatDeck script to configure LabJack devices, to generate and acquire signals from the electronic circuits as described above. ere, details about the configuration of the selected features in this experiment are explained. MatDeck also provides Graphical User Interface (GUI) plug-ins for simple, effective configuration. There are three plug-ins for three different groups of pins: ljdio_config_form() is used to configure DIOs, ljaio_config_form() is used to configure AINs and ljdac_config_form() is used to configure DACs. Here the details about the configuration of the selected features in this experiment are explained.

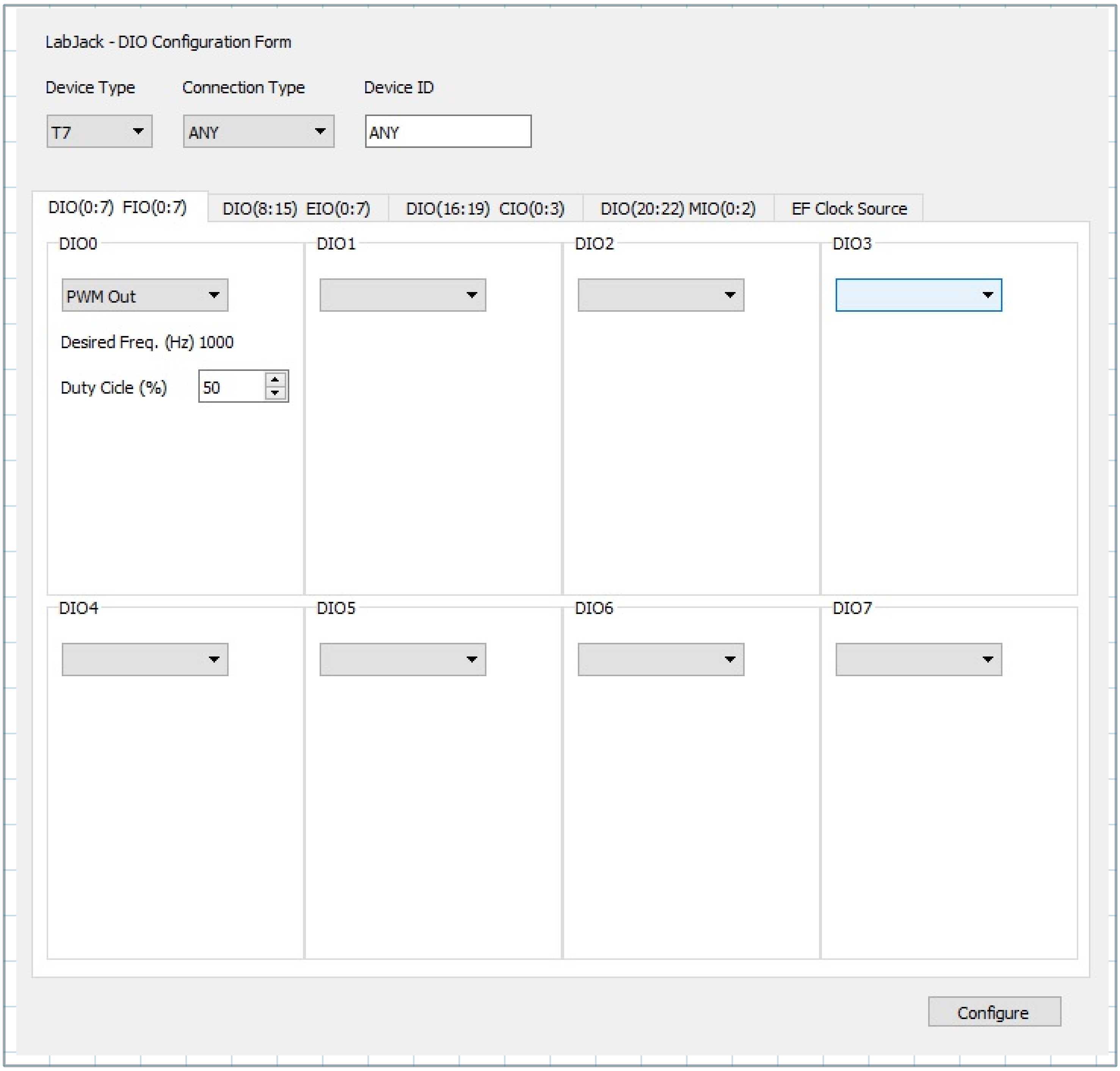
GUI Configuration of DIO EF PWM out

Here, DIO0 is used to produce IN2 signals. When DIO0 is high, the transistor is switched on and the current through the resistors heats the temperature sensor. If DIO0 is low, the transistor is switched off and there is no current, thus the temperature will fall.

GUI form for DIO configuration can be started using ljdio_config_form(), as follows. The form is embedded within the canvas and used for the configuration of DIOs.

```
ff := ljdioT7_config_form(0 , "DIO Form")
ljdioT7_config_form_configure(ff)
```

PWM Out at FIO0(DIO0) requires the clock source, thus the clock is first configured. There are three parameters to select for the configuration: clock source, clock divisor and the roll value for the given clock. There are three different clocks supported by T7, the most common is clock0 whose frequency is 80MHz. The clock divisor can be any power of two from 1, 2, up to 256, in this example we select value 1. The roll value is determined according to the desired frequency of the PWM Out signal. For example, if the desired frequency is 1kHz, the roll value is 80Hz/Divisor/1kHz=80000.In the GUI, it is possible to choose and set the desired frequency or desired roll value. PWM out at FIO0 (DIO0) is configured by selecting the appropriate option from the drop down menu. In the GUI, it is possible to set the desired value of the duty cycle directly to 50%.


LabJack - DIO Configuration Form

Device Type: T7
Connection Type: ANY
Device ID: ANY

DIO0: PWM Out
Desired Freq. (Hz) 1000
Duty Cicle (%) 50

GUI Configuring Temperature Measurement

The temperature is measured by using AIN2 channel of T7, where OUT2 signal is connected. The low power linear active thermistor circuit MCP9701A is used as a temperature sensor. Here, OUT2 is voltage that depends on the ambient temperature, which should be converted to the temperature using linear function given in data-sheet. The sensor transfer function is:

$$V_{OUT} = T_C \cdot T_A + V_{0°C}$$

Here, V_{OUT} is sensor output voltage, T_A is ambient temperature, T_C is temperature coefficient, and $V_{0°C}$ is sensor output voltage at 0°C. From MCP9701A datasheet, T_C=19.5 mV/°C and $V_{0°C}$=400mV. In order to determine the temperature from the voltage, we need the inverse function.

$$T_A = V_{OUT}/T_C - V_{0°C}/T_C$$

Slope and offset can be determined as follows:

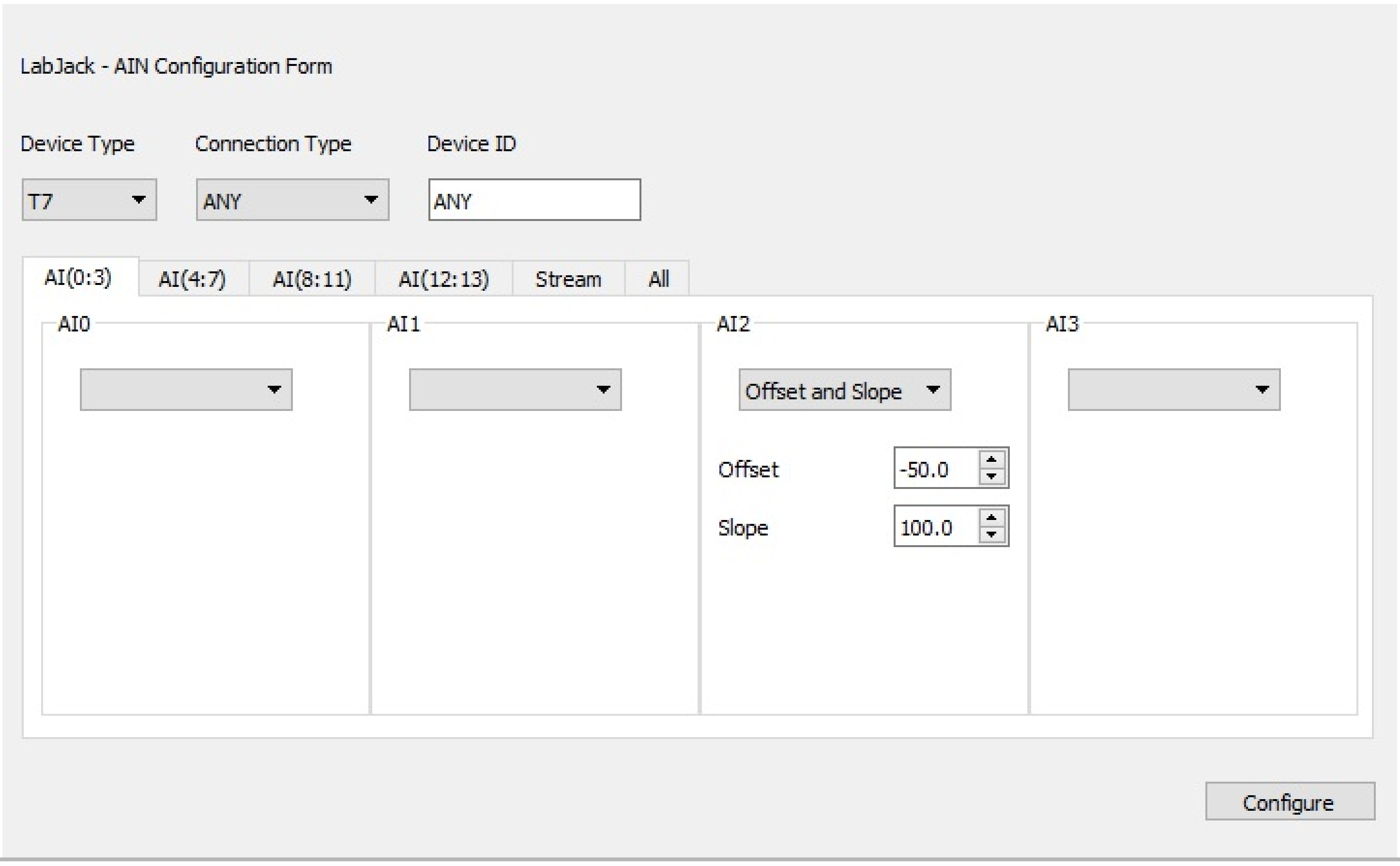
```
Tc := 0.0100
V0 := 0.5
Slope := 1 / Tc
Offset := -V0 / Tc
```

AIN2 is configured to use the Offset and Slope extended feature, EF_INDEX is 1, which automatically adds a slope and an offset to analog readings according to the linear function above.

$$Slope = 100 \qquad Offset = -50$$

MatDeck provides ljaio_config_form() which can be used to set all the parameters graphically, which is very convenient for the user. In the next segment, there is an illustration on how to use the AIN configuration form. At the beginning, the form is evoked by calling ljaio_config_form(). The form is embedded within the canvas and used for the configuration of AINs.

```
ff2 := ljainT7_config_form(0 , "AIN Form")
ljainT7_config_form_configure(ff2)
```



LabJack - AIN Configuration Form

Device Type	Connection Type	Device ID
T7	ANY	ANY

AI(0:3) | AI(4:7) | AI(8:11) | AI(12:13) | Stream | All

AI0	AI1	AI2	AI3
		Offset and Slope	
		Offset -50.0	
		Slope 100.0	

Configure

Use of Configured LabJack T7

In order to use the configuration and use the device, the LabJack T7 device should be opened in the document:

```
5 dev := ljdevice_open("any", "any", "any")
```

The temperature is automatically read using:

```
6 Ta := ljdevice_read(dev, "AIN2_EF_READ_A")
```

```
Ta = Ta    C
```

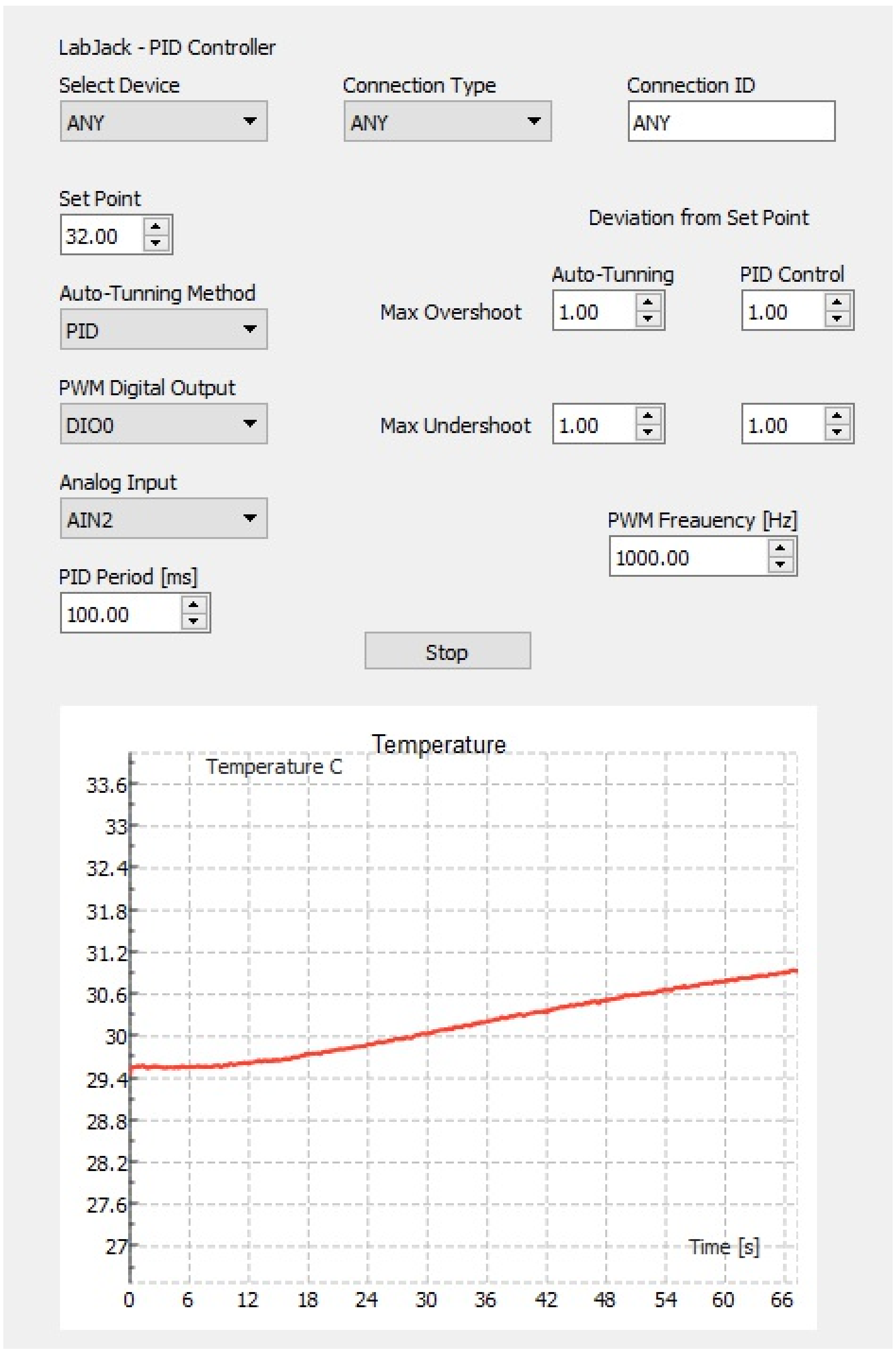
PID temperature control

MatDeck provides a PID controller which can adjust the ambient temperature close to the sensor by switching the transistor on and off in the circuits shown in the schematics. MatDeck PID controller is used in the real-time operation in the proposed heating system. PID controller is a three component controller having proportional, integral and derivative terms. A PID controller continuously calculates an error value as the difference between a desired setpoint temperature and a measured temperature from AIN2 LabJack channel, applies a correction based on proportional, integral, and derivative terms using control variables. In the proposed example the control variable is the duty cycle of the PWM out signal at DIO0. The user specifies the clock value which is the PID exchange period and the set point which is related to the controlled system. PID controller widget contains the graph where it is possible to see the value of the measured process variable versus the set point.

Auto-tuning PID controller is the adjustment of its control parameters (proportional gain, integral gain, derivative gain) to the optimum values for the desired control response. Ziegler–Nichols method for auto-tuning is the most common method in practice. The auto-tuning is based on the Ziegler–Nichols method, with the following different controllers: p, pi, pid, less overshoot, no overshoot, and Pessen integral.

```
f := ljpid_form(0 , "PID form")

ljpid_form_configure(f)
```



LabJack - PID Controller

Select Device: ANY

Connection Type: ANY

Connection ID: ANY

Set Point: 32.00

Auto-Tunning Method: PID

PWM Digital Output: DIO0

Analog Input: AIN2

PID Period [ms]: 100.00

Deviation from Set Point

	Auto-Tunning	PID Control
Max Overshoot	1.00	1.00
Max Undershoot	1.00	1.00

PWM Freauency [Hz]: 1000.00