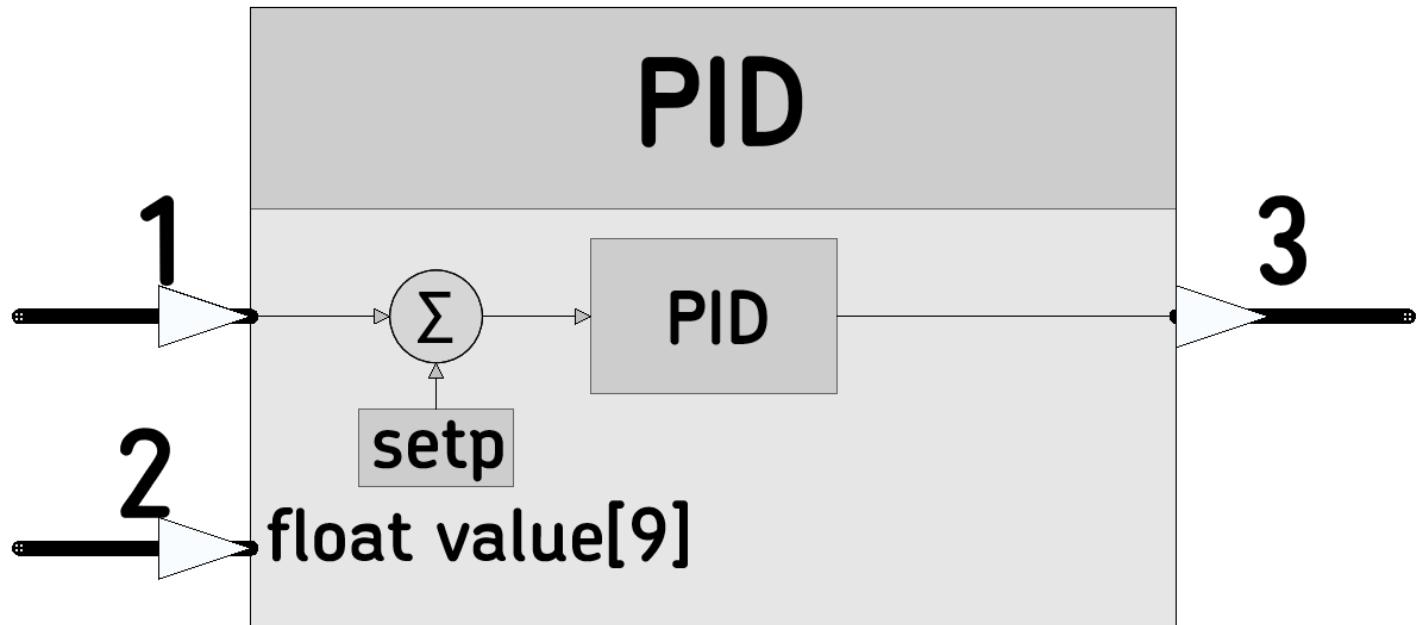


38D_PID_test

Testing program for the nBlocksStudio Node **PID**. The C++ source code, is automatically generated from the schematic Design [38D_PID_test_SCHEMATIC](#) . Controls a 40W 3D-printer heating element with PWM via power-MOSFET, using a Thermistor for temperature sensing.

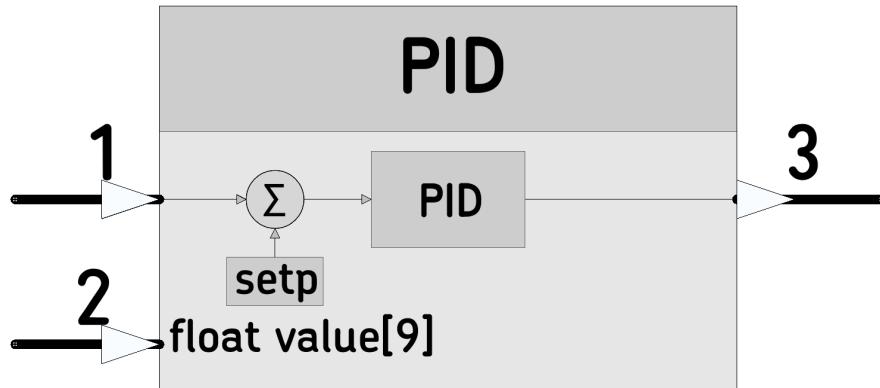
Node **PID**



Kp	0.2	minOutput	0.0
Ki	0.025	maxOutput	1.0
Kd	0.0	PIDmode	AUTOMATIC
stime	0.2	PIDdirection	DIRECT
		setup	60.0

Node PID details

PID controller calculates an `error` value as the difference between a measured `Input` and a desired `setpoint`. The controller attempts to minimize the `error` by adjusting an `Output` (source Wikipedia)



- Category: Function
- HAL: No Hardware dependencies
- Status: Node defined and Compiled
- Author: Nikolaos Chalikias

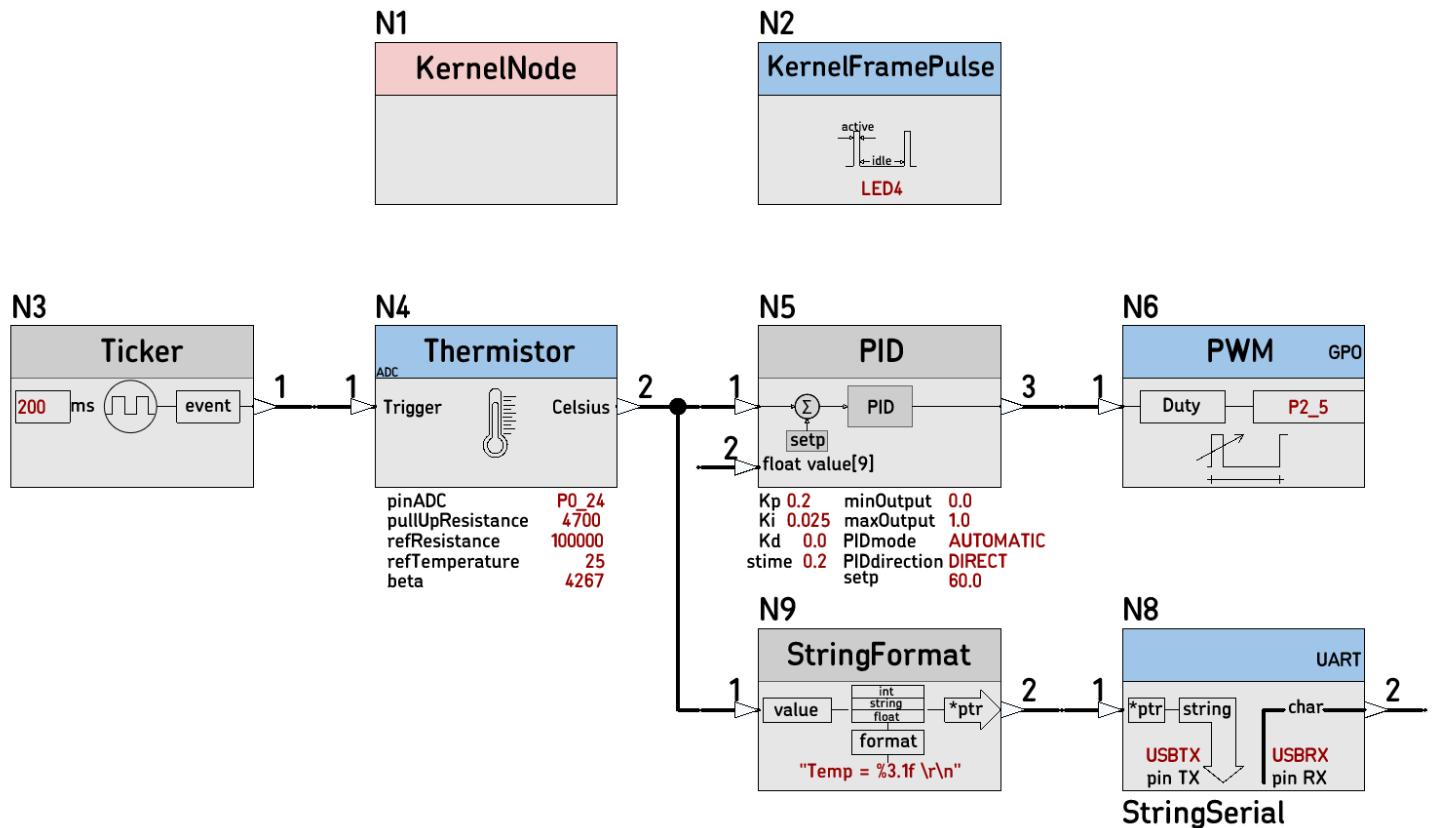
Inputs/Outputs

- (schematic pin 2) float: `Input` (sensed) value
- (schematic pin 1) Input: All PID parameters in form of array `* float : 9 values: [Kp, Ki, Kd, stime, minOutput, MaxOutput, PIDmode(0.0 or 1.0), PIDdirection (0.0 or 1.0), Setpoint]`. Overrides the default values
- (schematic pin 3) `uint32_t`: `Output` (controlling) value, normalized to `0xFFFF`

Parameters

- float: `Kp` default
- float: `Ki` default
- float: `Kd` default
- float: `stime`: sampling time in sec: default
- float: `minOutput`: default
- float: `maxOutput`: default: in current implementation is limited internally to ≤ 1
- `PIDMode`: MANUAL, AUTOMATIC: default
- `PIDDirection`: DIRECT, REVERSE: default
- float: `setp`: Setpoint: default

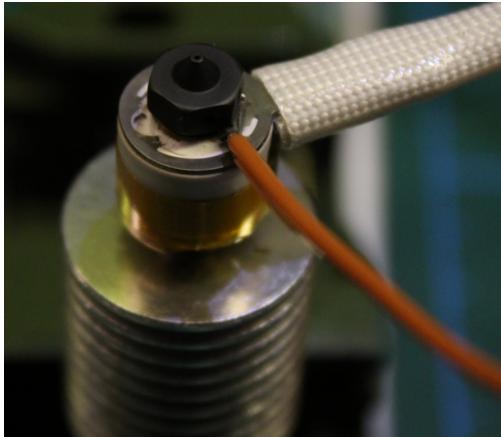
nBlocksStudio Schematic Design



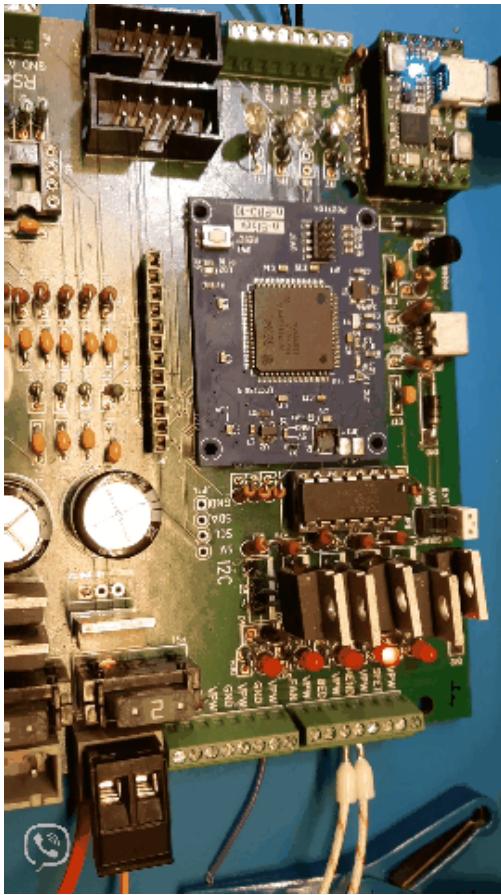
- N1: The KernelNode is configured for mbed
- N3: The ticker is configured to trigger the sensing / control loop, every 200ms
- N4: The thermistor Node is configured for:
 - Microcontroller ADC-pin to be used for measuring the Thermistor voltage.
 - The Thermistor characteristics and Bias
- N5: The PID Node configured with: PID-Coefficients & Temperature set at 60 °C
- N6: The PWM Node drives the Heating ellement via a Power-Mosfet from Microcontroller pin P2.5
- N9 and N8 are used for dedugging / Monitoring via USB PC connection and a serial Terminal.

Setup

Hot-End and an external thermocouple probe for temperature validation (red cable)

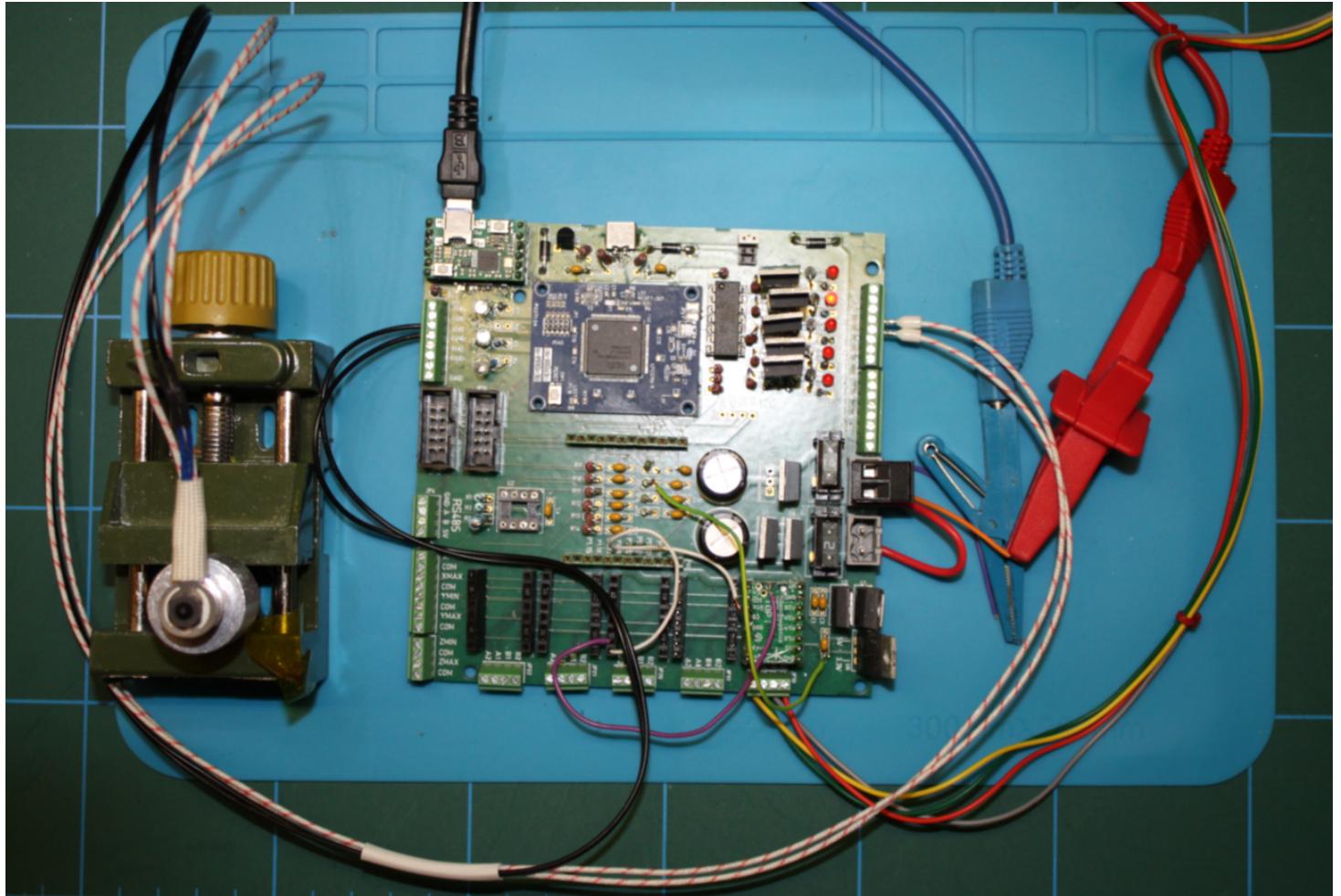


The PID test program runs in a n-3DP board, with sampling rate 200ms, driving the Hot-End



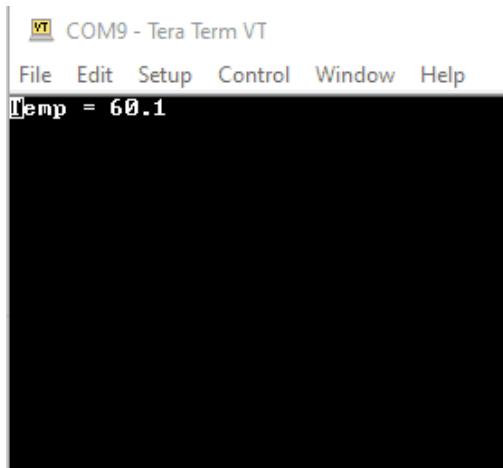
n-3DP motion control/3D printer motherboardc with n-PRO-10 LPC1768 microcontroller board.

- LPC1768 PWM output drives the power MOSFET which powers the Hot-End (white/red pair of cables)
- The Thermistor is connected to the terminal Block, filtered and then connected to an LPC1768 ADC input (black pair of cables)



Temperature control results

Top: Stabilized temeprature at 60 °C measured by Thermistor



VT COM9 - Tera Term VT
File Edit Setup Control Window Help
Temp = 60.1

Middle: Temeperature measured with an external Thermocouple probe



Bottom: n-3DP supplied with 19V and the the Current consumption is stabilized at 135 mA (Power = 2.5W)



Thermistor Node

Thermistor Node with an added debugging printf(), prints the ADC value for the temperature value it exports.

```
28 < void nBlock_Termistor::endFrame() {
29   << if (_read_requested) {
30     << _read_requested = 0;
31
32     // This is a normalized value respective to Vdd
33     // (in other words, adc_read=0.3 means 0.3*Vdd)
34     float adc_read = _adc.read();
35     printf("adcread= %f-->\n", adc_read);
36     // Relative current = I / Vdd (normalized to Vdd)
37     // It is the current in [A] if Vdd was 1.0V
38     float relative_current = (1.0-adc_read)/_pullup;
39     float therm_R = adc_read / relative_current;
40
41     // This node uses the beta equation:
42     // 1/T = 1/T0 + (1/β) . ln (R/R0)
43     float inv_T = _inv_ref_temp + _inv_beta * log(therm_R / _ref_res);
44     float result = (1.0/inv_T) - T_K;
45
46     output[0] = PackFloat(result);      //output[0] = PackFloat(result/2.5);
47     available[0] = 1;
48   }
49 }
50 }
```

```
VT COM9 - Tera Term VT
File Edit Setup Control Window Help
adcread= 0.826129Temp = 59.9
adcread= 0.825885Temp = 59.8
adcread= 0.825885Temp = 59.9
adcread= 0.826129Temp = 59.9
adcread= 0.826129Temp = 59.8
adcread= 0.826129Temp = 59.8
adcread= 0.825885Temp = 59.8
adcread= 0.826129Temp = 59.9
adcread= 0.825885Temp = 59.8
adcread= 0.825885Temp = 59.9
adcread= 0.826129Temp = 59.9
adcread= 0.825885Temp = 59.8
adcread= 0.826129Temp = 59.9
adcread= 0.826129Temp = 59.8
adcread= 0.826129Temp = 59.8
adcread= 0.825885Temp = 59.8
adcread= 0.825885Temp = 59.9
adcread= 0.826129Temp = 59.9
adcread= 0.825885Temp = 59.8
adcread= 0.826129Temp = 59.9
adcread= 0.825885Temp = 59.8
```

The formula used in the Node C++ code is validated and confirmed using an excel spreadsheet and online Thermistor calculators. The theoretical Temperature value for the corresponding ADC measurement is precise. Measuring with the Multimeter indicates a slight difference from the measured voltage from ADC.

Conclusion on Thermistor Node precision

The Thermistor Node works good, the ADC Hardware front-end needs some improvement, but the precision is still good for a hot-end temperature control.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1		T_K	PULLUP	REFRES	refTemp_celsius	REFTEMP	BETA	INV_REF_TEMP	INV_BETA	ADC_READ	REL_CURRENT=(1.0- adc_read)/_pullup;	THERMISTOR_R = inv_T=_inv_ref_temp + _inv_beta*log(therm_R/ _ref_res);	result = (1.0*inv_T) - T_K;							
2		273	4700	100000	25	298	4267	0.003355705	0.000234357	0.1	0.000191489	522.222222	0.0021242	470.7655534	197.7655534					
3		273	4700	100000	25	298	4267	0.003355705	0.000234357	0.5	0.000106383	4700	0.002639134	378.9121906	105.9121906					
4		273	4700	100000	25	298	4267	0.003355705	0.000234357	0.8664	2.84255E-05	30479.64072	0.003077263	324.9641124	51.9641124					
5		273	4700	100000	25	298	4267	0.003355705	0.000234357	0.8255	3.71277E-05	22234.09742	0.003003339	332.9627126	53.9627126					
6		273	4700	100000	25	298	4267	0.003355705	0.000234357	0.95510984	9.5511E-06	100000.0055	0.003355705	297.9999983	24.9999985					
7																				
8																				
9																				
10	void nblock_Thermistor::endFrame() {																			
11	if (_read_requested) {																			
12	_read_requested = 0;																			
13	}																			
14	// This is a normalized value respective to Vdd																			
15	// (in other words, adc_read=0.3 means 0.3*Vdd)																			
16	float adc_read = _adc.read();																			
17	// Relative current = I / Vdd (normalized to Vdd)																			
18	// It is the current in [A] if Vdd was 1.0V																			
19	float relative_current = (1.0*adc_read)/_pullup;																			
20	float therm_R = adc_read / relative_current;																			
21	// This node uses the beta equation:																			
22	// 1/T = 1/T0 + (1/B) * ln (R/R0)																			
23	float inv_T = _inv_ref_temp + _inv_beta * log(therm_R/_ref_res);																			
24	float result = (1.0/inv_T) - T_K;																			
25	output[0] = PackFloat(result);																			
26	available[0] = 1;																			
27	}																			
28																				
29																				
30																				
31																				
32																				
33																				
34																				
35																				
36																				
37																				
38																				
39																				
40																				

<https://www.electro-tech-online.com/tools/thermistor-resistance-calculator.php>

<https://www.giangrandi.org/electronics/ntc/ntc.shtml>

NTC Thermistor Resistance Calculator V2.0

Enter Thermistor Datasheet Values.

Rnew=R0*exp-B*((1/273+Tref)-(1/273+Tnew))

Thermistor Beta Value: 4267

Resistance at Ref Temperature, Ohms: 100000

Reference Temperature, Cdeg: 25

Target Temperature, Cdeg: 52

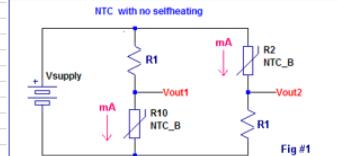
Thermistor Resistance Value at Target Temperature

Resistance Value at 52 Cdeg is: 30470 Ohms.

[Click to Calculate Thermistor Resistance](#)

Supply Voltage: 3.173

R1. Series Load Resistor, Ohms: 4700



Thermistor Resistance Junction Voltage Values for Fig #1

Vout1= 2.75V... Vout2= 0.42V. NTC current=0.09mA

[Click to Calculate the Junction Outputs](#)

NTC parameters:

R25 = 100000 Ω

β = 4267 K

Temperature and resistance:

R = 30470 Ω

T = 52 °C

[Calculate T](#) [Calculate R](#)

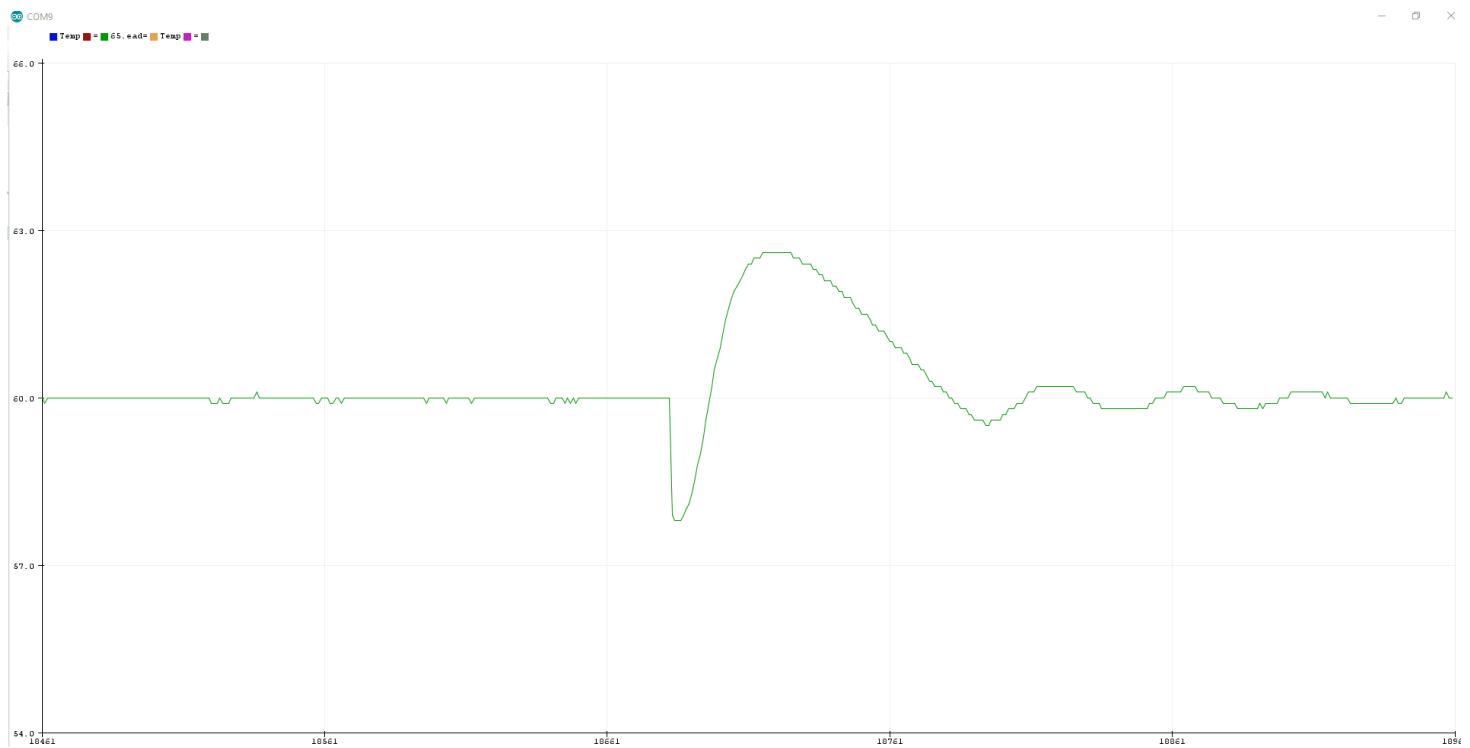
PID control evaluation

```
nBlock_PID nb_nBlockNode3_PID [0.2, 0.025, 0.0, 0.2, 0.0, 1.0, AUTOMATIC, DIRECT, 60.0];
```

After trial and error, the values below result in an acceptable controlling performance.

- P = 0.2
- I = 0.025
- D = 0
- SamplingTime = 0.2 sec
- minOutput = 0
- maxOutput = 1
- setPoint = 60 °C

Powered-on at a temperature 40 °C, the PID controller tries to fix the temperature to the 60 °C set point fast, so we have an overshoot, then the system is stabilized with a slight oscillation around the 60 °C value.



Plotted with Arduino plotter.

Disturbing the heat balance, by *just slightly-blowing air for 1 sec*, to the hot-end.
The Sensing and PID system is sensitive enough, to detect and correct the deviation.

