Temperature Regulation with PID Control

Okay, simple, this is my temperature control system with PID algorithm, not sure if it can help you but I’m sure you will study something. Although this is a cheapy system but it can control ±0.5℃ which pretty well.

My first rule is: As simple as you (I) can

What is temperature control?

You want the temperature to fluctuate within the range you desire. Inside a room, heating relate machine. For more information, just GG it.

What is PID Control?

PID Control stands for Proportional-Integral-Derivative Control. It's a control loop feedback mechanism widely used in industrial control systems to regulate processes. Here's what each component does:

Proportional (P): The control action is directly proportional to the error signal, which is the difference between the desired setpoint and the actual process variable. This component provides an immediate response to changes in the system.

Integral (I): The integral component sums up the error over time and corrects the steady-state error. It helps to eliminate any residual error that remains after the proportional control has brought the system close to the setpoint.

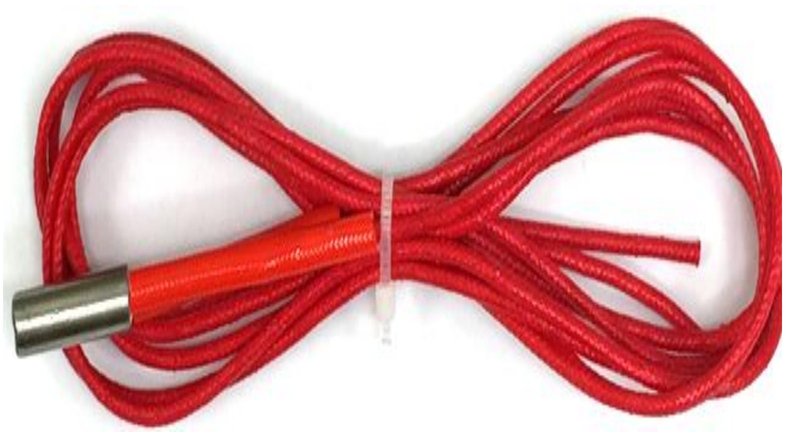
Derivative (D): The derivative component predicts future behavior of the system based on the rate of change of the error signal. It helps to anticipate and dampen rapid changes, thus improving system stability and response time. This link (Just 5 mins) demonstrates PID pretty well, I recommend you watch it.

https://www.youtube.com/watch?v=4Y7zG48uHRo&list=PL9eGdpR5-_y7ltkWsbtxaJFL0DBEU8oYI&index=7

My system:

My system will control a temperature around 39 to 40 degrees Celsius, which is 39.5 ± 0.5℃ , with 12V heater, and 5V for sensor and controller.

Heater: Just a simple heater in 3D printer, so it's very fluctuate.



SSR: a solid state relay definitely need for this, normal relay can't turn on and off at high frequency, plus it's soundless, and you will need a heat sink if you don't want your SSR broken in about 5 mins. Check it frequently.

Controller ATmega328P: It’s Arduino Uno

Temperature sensor PT100 with module MAX31865: If you want precisely and archive this goal, it’s recommend to use high quality sensor, this PT100 is resistor type so when temperature change, it’s resistance change, we can measure it change rate and convert to voltage which is readable, to do so, MAX31865 come into part, It's specifically designed to convert the resistance variations of the PT100 sensor into a digital signal that can be read and processed directly by a microcontroller without any further ado.

A 12V fan with a SSR too: To reduce temperature when overheat

Module LM2596: I use this to generate 3.3V for Max31865. You have to adjust screw on it to get output 3.3V. Depend on your system, power supply and be set up in some different way, you shouldn't use power supply for both controller and driving system (motor, heater, …) because it lead to noise and other dangerous. This is experiment test so I used one adapter for both :)

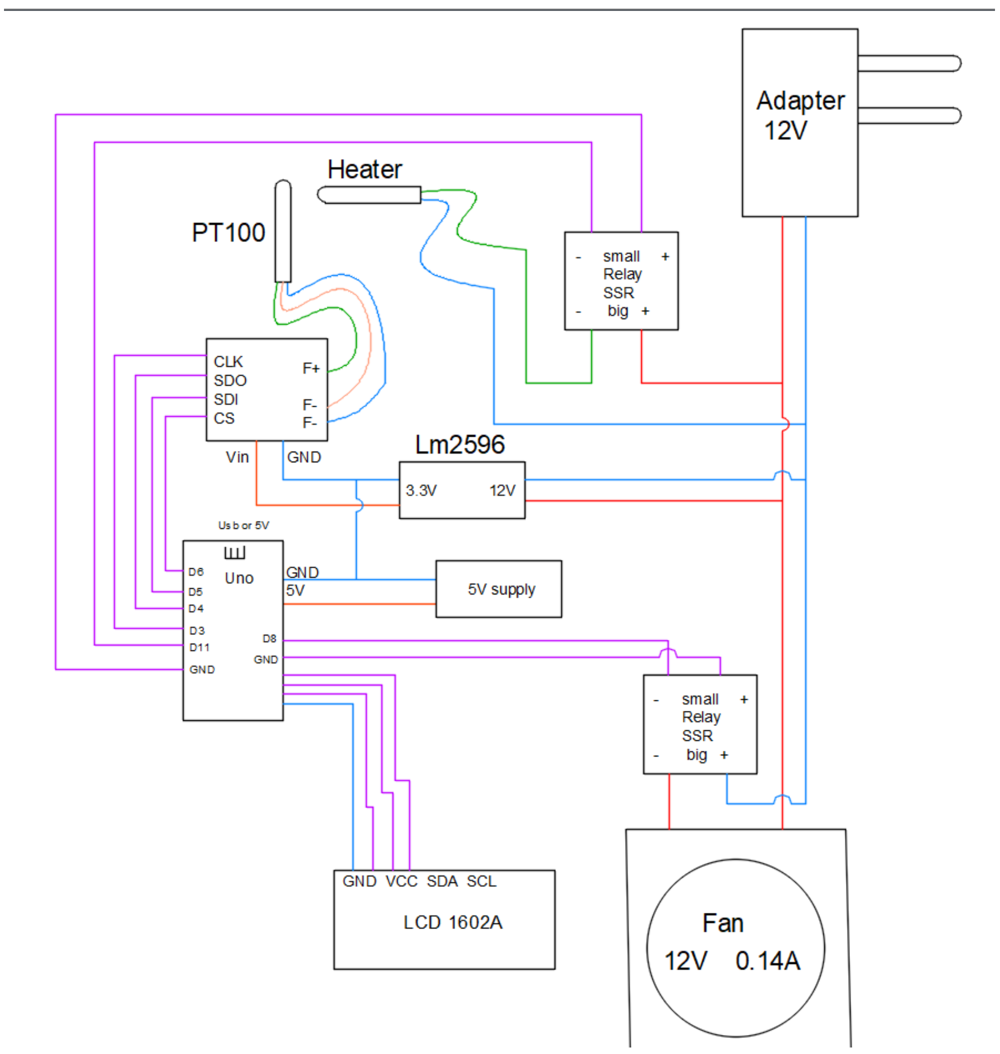
An 12V adapter, a plug, and wires to transmit electrical all over system.

You will need a box to control temperature in it due to my big test. If the system is small, tie the sensor and the heater in a fixed range or you will get different result.

If you have LCD, just do it.

How to start?

Well, you will have to do many things to setup, basically this is wiring schematic.



Note:

- Connect GND of MAX31865 with controller GND to sync data.
- No matter positive and negative of SSR as long as you understand SSR, but you should follow label on it.
- You have to solder something in MAX31865. Solder closed the jumper labeled 2/3. Then solder closed mid and right of "24 3" jumper. (4 n 3)
- LCD (optional) you will have module SPI or Arduino don't have enough pins.

How to implement and coding?

Step 1:

Firstly, I assume you have some knowledge about Matlab – Simulink and Arduino, because this writing will double if you don't, and I'm too lazy for this. Although, I will give some basic steps to sync with. And this is some key words that you may want to know:

- Connect Arduino with Matlab (Send n receive data)
- Communicate between Matlab and Simulink (Send n receive data)
- How to use System Identification to obtain transfer model

Step 2:

Arduino:

Just download Arduino IDE, it's free, add libraries, setup board, port, make it compile code, finish. Arduino will connect to Matlab through a port, in my case, COM5 with baud rate 115200 (it's default in Matlab).

Make a code that send a PWM value constantly to the SSR of heater, by this way we will obtain first raw transform function. I will make a value of 170 (min – max = 0 – 255), to get it, just connect to Matlab and read temperature. Save it to an array.

Arduino code: Basically this code will send PWM electrical value to pin D11 which activate the SSR of heater. PWM value read from the cable connected to Matlab.

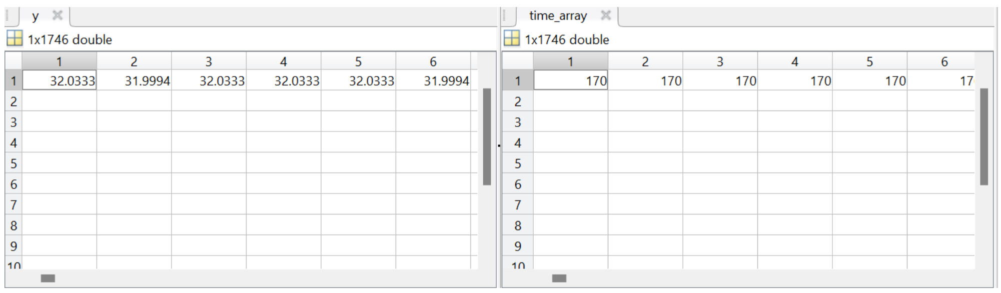
```
#include <Adafruit_MAX31865.h>

// Use software SPI: CS, DI, DO, CLK
Adafruit_MAX31865 thermo = Adafruit_MAX31865(6, 5, 4, 3);
double input, output;    // Current input and output
// The value of the Rref resistor. Use 430.0 for PT100 and 4300.0 for PT1000
#define RREF      430.0
// 100.0 for PT100, 1000.0 for PT1000
#define RNOMINAL  100.0
int pwmval = 0;
const int PWM = 11;

void setup() {
  Serial.begin(115200, SERIAL_8N1); // Set the baud rate to match the MATLAB
script
  pinMode(PWM, OUTPUT);
  thermo.begin(MAX31865_2WIRE);  // set to 2WIRE or 4WIRE as necessary
}

void loop() {

  input = thermo.temperature(RNOMINAL, RREF);

  if (Serial.available() > 0)
  {
    pwmval = Serial.parseInt();
    Serial.println(input, 6); // 6 decimal places precision
  }

  analogWrite(PWM, pwmval);

// Print out for debug
//  Serial.println(pwmval);
//  Serial.println(input);
// The delay is pretty slow if you ask
  delay (500);

}
```

Step 3:

Matlab code: You have to define port (`%s = serialport("COM5", 115200)`) before run this script.

```
clc
%s = serialport("COM5", 115200)
receivedDataArray = [];
% Open the serial connection
fopen(s);
try
    while true

    % Send integer value to Arduino
    dataToSend = 170; % Replace with your integer value
    fprintf(s, '%d]', dataToSend);

    % Read the double value from Arduino
    data = fscanf(s, '%f');

    % Display the received data
    disp(['Received data from Arduino: ' num2str(data)]);

    % Do further processing with the data if needed
    receivedDataArray = [receivedDataArray data];

    % Store sensor value in array for later analysis or plotting
    % Wait for a moment to let Arduin\ o process the data
    pause(0.5);
     end
catch
    % Close the serial connection when an error occurs or when you stop the script
    fclose(s);
    delete(s);
    clear s;
    disp('Serial connection closed');
end
```

Then, in Matlab, make a time array match every value you have read, this is the input value, example, you have 1800 value. create an array with dimensions 1x1800, where each column contains a value of 170. Once you have input and output (temperature) you can use function in Matlab to get your raw transform function.



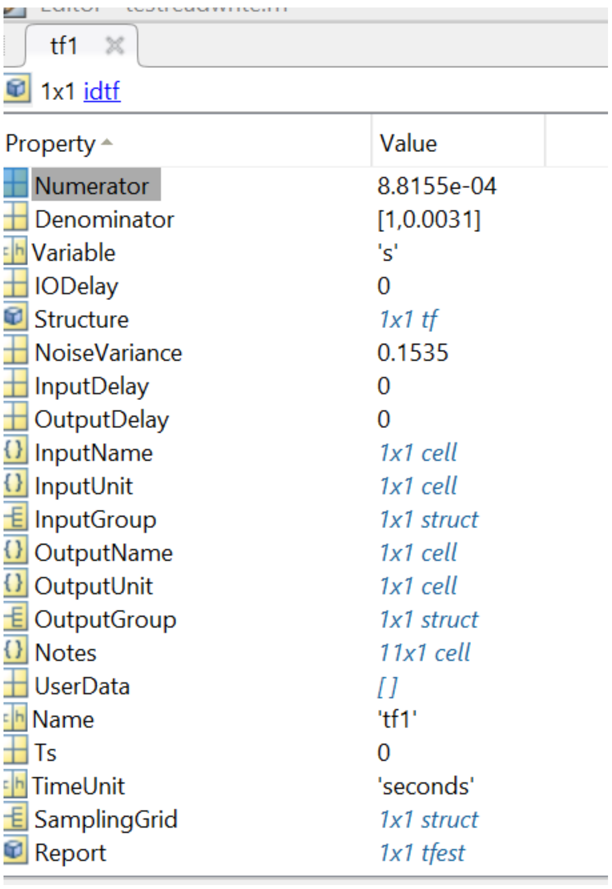
y
1x1746 double

	1	2	3	4	5	6
1	32.0333	31.9994	32.0333	32.0333	32.0333	31.9994
2						
3						
4						
5						
6						
7						
8						
9						
10						

time_array
1x1746 double

	1	2	3	4	5	6
1	170	170	170	170	170	17
2						
3						
4						
5						
6						
7						
8						
9						
10						

Open System Identification, use Time Domain Signals, insert input and ouput array, adjust your start time and sample time, use option Transfer Function Model then you will have transfer function “tf1”, export to workspace.



tf1

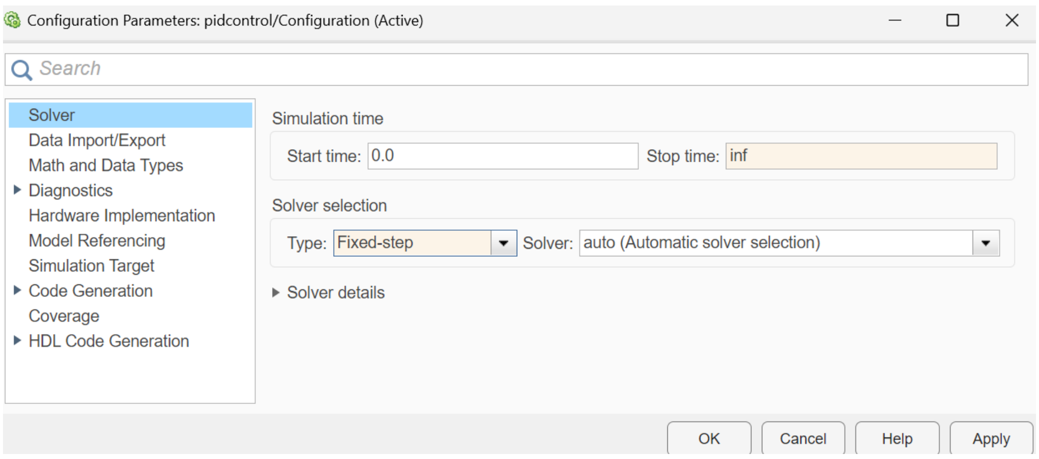
1x1 idtf

Property	Value
Numerator	8.8155e-04
Denominator	[1,0.0031]
Variable	's'
IODelay	0
Structure	1x1 tf
NoiseVariance	0.1535
InputDelay	0
OutputDelay	0
InputName	1x1 cell
InputUnit	1x1 cell
InputGroup	1x1 struct
OutputName	1x1 cell
OutputUnit	1x1 cell
OutputGroup	1x1 struct
Notes	11x1 cell
UserData	[]
Name	'tf1'
Ts	0
TimeUnit	'seconds'
SamplingGrid	1x1 struct
Report	1x1 tfest

Step 4:

Matlab – Simulink:

I use Matlab R2023b, download, make a blank Simulink model, and manage to connect Arduino with Matlab, ensure these setting to run in **Hardware mode**. (Sorry I'm not sure if it need these settings)



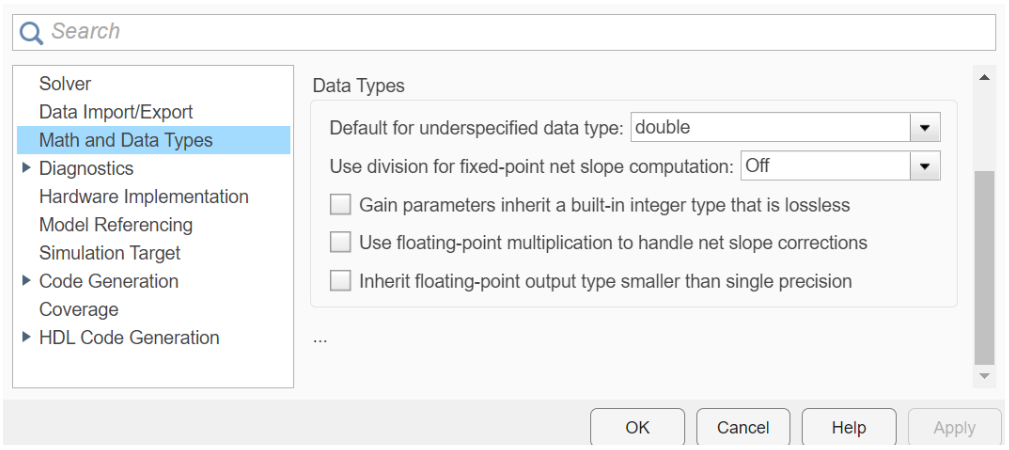
Configuration Parameters: pidcontrol/Configuration (Active)

Simulation time

Start time: 0.0 Stop time: inf

Solver selection

Type: Fixed-step Solver: auto (Automatic solver selection)



Data Types

Default for underspecified data type: double

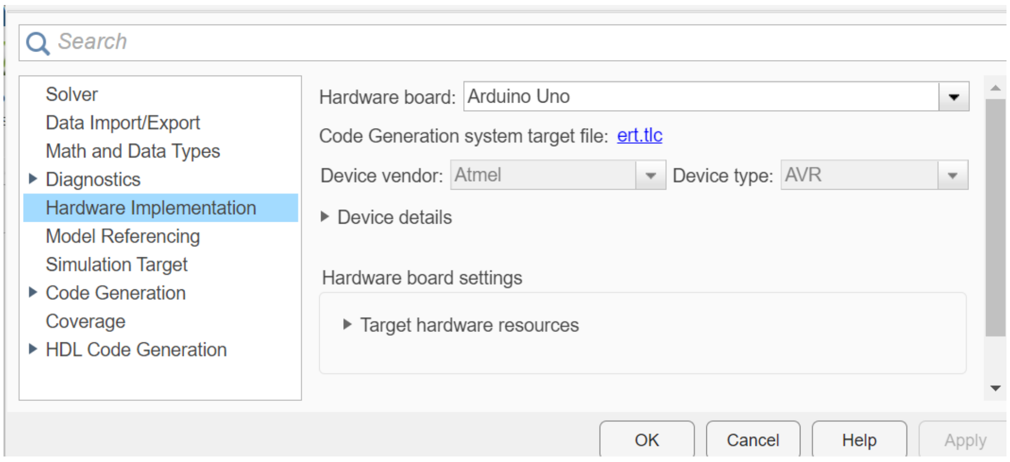
Use division for fixed-point net slope computation: Off

☐ Gain parameters inherit a built-in integer type that is lossless

☐ Use floating-point multiplication to handle net slope corrections

☐ Inherit floating-point output type smaller than single precision

...



Hardware board: Arduino Uno

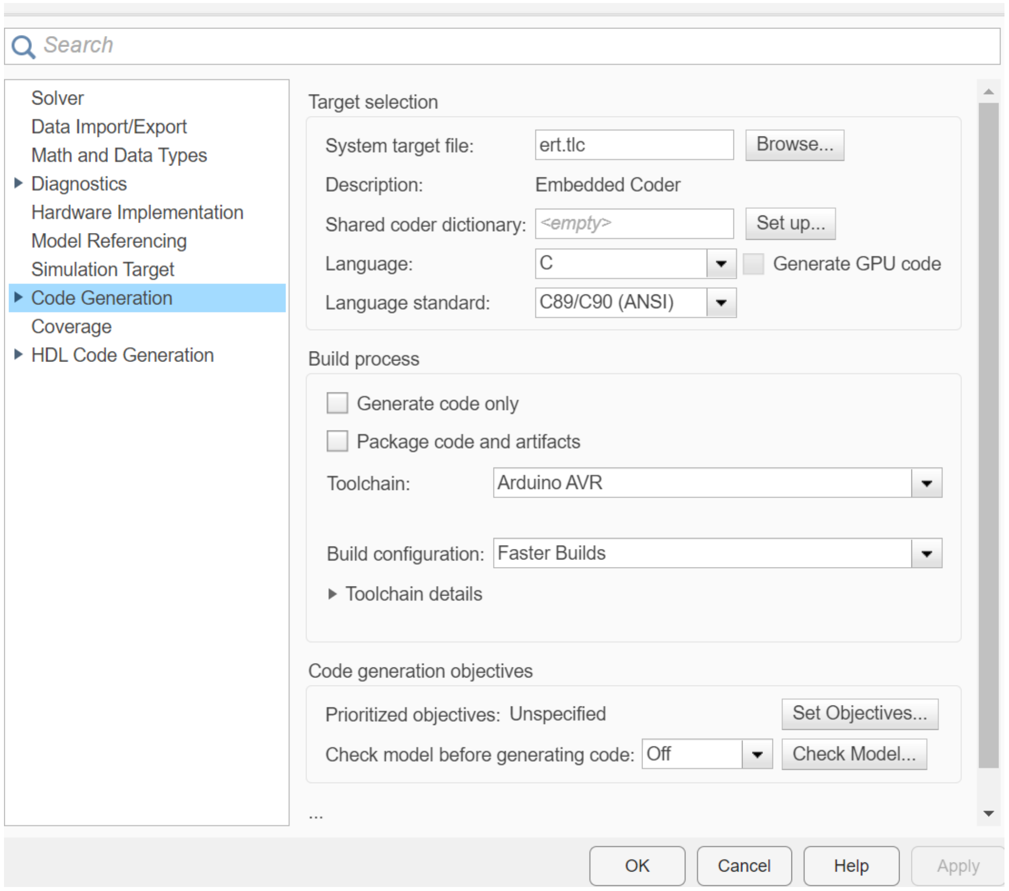
Code Generation system target file: ert.tlc

Device vendor: Atmel Device type: AVR

Device details

Hardware board settings

Target hardware resources



Target selection

System target file: ert.tlc

Description: Embedded Coder

Shared coder dictionary: <empty>

Language: C ☐ Generate GPU code

Language standard: C89/C90 (ANSI)

Build process

☐ Generate code only

☐ Package code and artifacts

Toolchain: Arduino AVR

Build configuration: Faster Builds

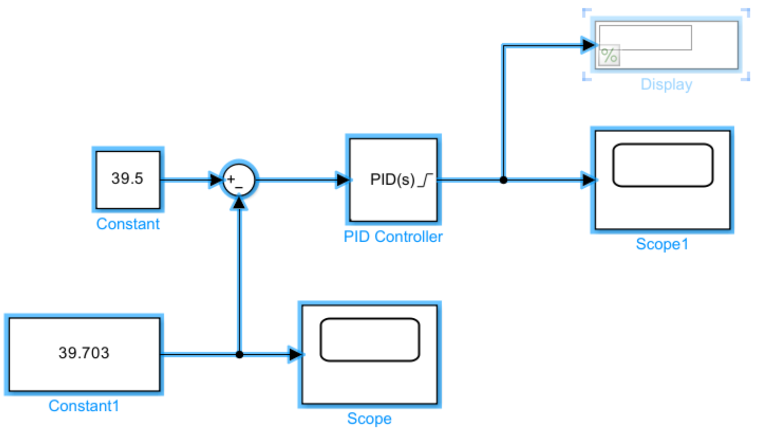
Toolchain details

Code generation objectives

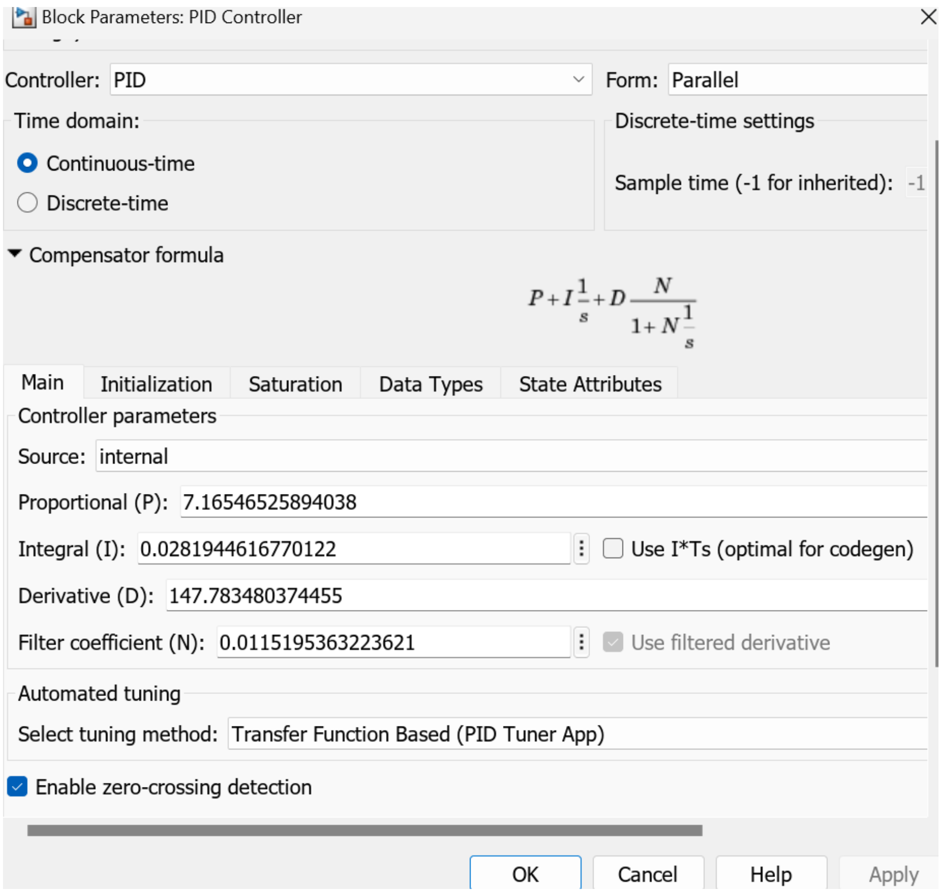
Prioritized objectives: Unspecified

Check model before generating code: Off

...



Then open PID block, adjust these setting:


Block Parameters: PID Controller

Controller: PID   Form: Parallel

Time domain:
[✓ Continuous-time]
Discrete-time

Discrete-time settings
Sample time (-1 for inherited): -1

Compensator formula

$$P+I\frac{1}{s}+D\frac{N}{1+N\frac{1}{s}}$$

Main | Initialization | Saturation | Data Types | State Attributes

Controller parameters

Source: internal

Proportional (P): 7.16546525894038

Integral (I): 0.0281944616770122   ☐ Use I*Ts (optimal for codegen)

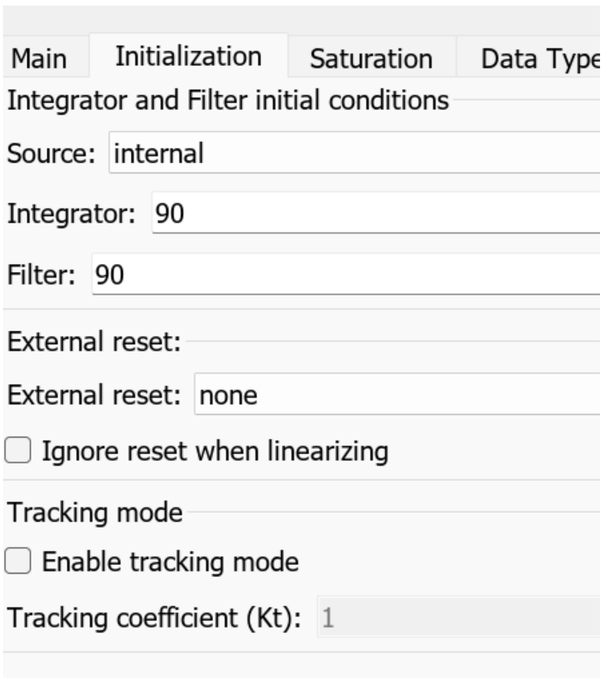
Derivative (D): 147.783480374455

Filter coefficient (N): 0.0115195363223621   ✓ Use filtered derivative

Automated tuning

Select tuning method: Transfer Function Based (PID Tuner App)

✓ Enable zero-crossing detection


Main | Initialization | Saturation | Data Type

Integrator and Filter initial conditions

Source: internal

Integrator: 90

Filter: 90

External reset:

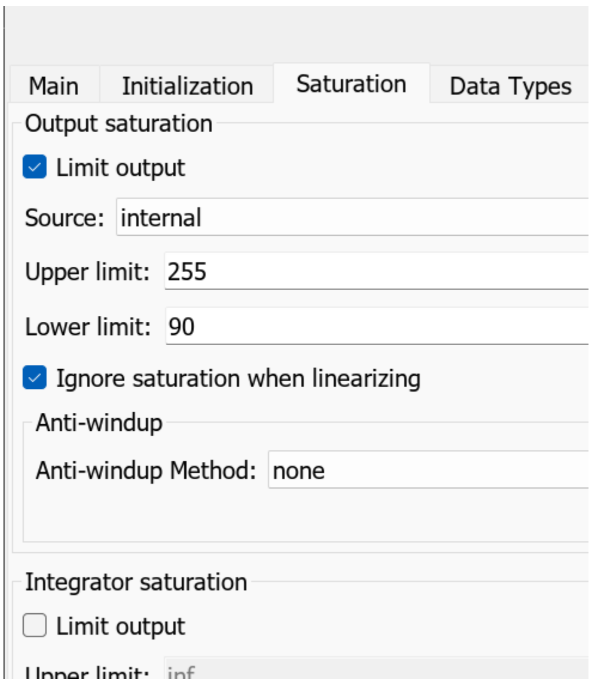
External reset: none

☐ Ignore reset when linearizing

Tracking mode

☐ Enable tracking mode

Tracking coefficient (Kt): 1


Main | Initialization | Saturation | Data Types

Output saturation

☑ Limit output

Source: internal

Upper limit: 255

Lower limit: 90

☑ Ignore saturation when linearizing

Anti-windup

Anti-windup Method: none

Integrator saturation

☐ Limit output

Upper limit: inf

Note: If your heater does not respond linearly to changes in PWM (Pulse Width Modulation), it's essential to establish a lower limit. This is crucial because at low PWM values such as 20 or 30, the heater may not receive sufficient electricity to operate, the result is temperature will not change a bit. Consequently, it may require multiple attempts to determine the PWM range at which the heater begins to effectively warm the room. In my case, this threshold is 90. For upper limit, if you use 0 – 255 PWM variant, ofc set it to 255.

Step 5:

Now It's time to find real transfer function, It's easy as pie. Select PID Controller block in Simulink, set it to Autotune use PID Tuner, import the plant "tf1", then adjust 2 numbers on the top until it match your desire, click on "Show parameters" to see OverShoot and Setting time. Once finished, click "Update Block" and close it, your parameters will be set.

Step 6:

We have done the setup, now make real Arduino code and Matlab code for PID tuning, the Arduino read signals from Matlab which have been tuned by temperature datas and then paste it to heater.

As I mentioned, turn on the fan if It's about to overheat, you can have LCD if you want (this is SPI code).

As for Matlab code, remember to connect Arduino board to Matlab, define serial port if workspace not yet initialized, open the port, define some parameters, then in the while loop simply read and write data to Arduino.

After stop, you will have temperature array "y[]", do plot(y) if you want to see the result. (I'm sure you will)

Arduino code:

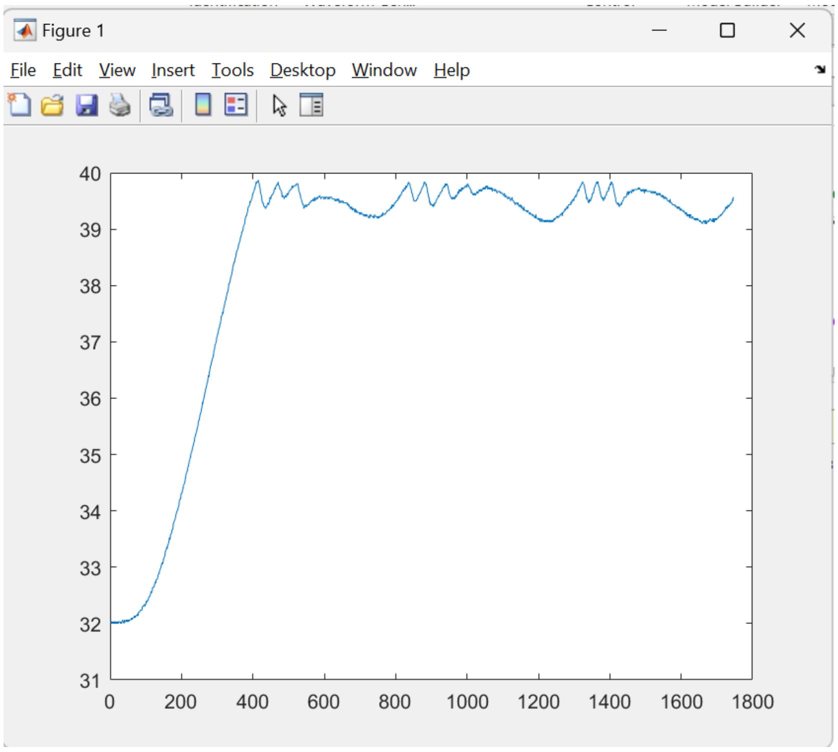
```
#include <Adafruit_MAX31865.h>

// Use software SPI: CS, DI, DO, CLK
Adafruit_MAX31865 thermo = Adafruit_MAX31865(6, 5, 4, 3);
// use hardware SPI, just pass in the CS pin
//Adafruit_MAX31865 thermo = Adafruit_MAX31865(10);
double input, output;    // Current input and output
int delayTime = 0;
// The value of the Rref resistor. Use 430.0 for PT100 and 4300.0 for PT1000
#define RREF      430.0
// The 'nominal' 0-degrees-C resistance of the sensor
// 100.0 for PT100, 1000.0 for PT1000
#define RNOMINAL  100.0
int pwmval = 0;
const int PWM = 11;

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0X27,16,2); //SCL A5 SDA A4
byte degree[8] = {
  0B01110,
  0B01010,
  0B01110,
  0B00000,
  0B00000,
  0B00000,
  0B00000,
  0B00000
};

void setup() {
  Serial.begin(115200, SERIAL_8N1); // Set the baud rate to match the MATLAB
script

  pinMode(PWM, OUTPUT);
  pinMode(8, OUTPUT);

  thermo.begin(MAX31865_2WIRE);  // set to 2WIRE or 4WIRE as necessary

  lcd.init();
  lcd.backlight();
}

void loop() {

  input = thermo.temperature(RNOMINAL, RREF);

  if (Serial.available() > 0)
  {
```

Matlab code

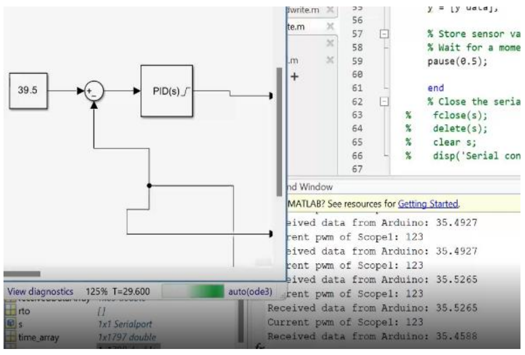
```
clc
% Define port
% s = serialport("COM5", 115200)

% Open the serial connection
fopen(s);
y = [];
i = 0;
data = 0;
% open_system('pidcontrol');
set_param('pidcontrol','BlockReduction','off');
set_param('pidcontrol','StopTime','inf');
set_param('pidcontrol','SimulationMode','normal');
set_param('pidcontrol','StartFcn','1');
set_param('pidcontrol','SimulationCommand','start');
% simOut = sim('pidnhietdo_new');
block = 'pidcontrol/Scope1';
fprintf(s, '%d]', 255);
pause(5);

while true

    % Send integer value to Arduino
    % dataToSend = get_param(['pidcontrol' '/PID Controller'],'Controller');
    % Replace with your integer value
    rto = get_param(block, 'RuntimeObject');

    dataToSend = round(rto.InputPort(1).Data);

    % Adjust data base on your system
    % if data >= 39.85
    %     dataToSend = 0;
    % elseif data < 39.85 && data >= 39.4
    %     dataToSend = dataToSend - 10;
    %     i=1;
    % elseif data < 39.4 && data > 39 && i==1
    %     dataToSend = dataToSend + 10;
    %
    % end
    % if data < 39.4 && data > 38 && i==0
    %     dataToSend = dataToSend - 20;
    % end
    % if data < 39.6 && data > 39.4
    %     dataToSend = dataToSend - 5;
    % end
    %dataToSend = get_param('pidcontrol/Scope1', 'OutputSignal')

    fprintf(s, '%d]', dataToSend);
    %disp(['pwm: ' num2str(get_param(['pidcontrol' '/PID Controller'],'Control-
ler'))]);
    disp(['Current pwm of Scope1: ' num2str(dataToSend)]);
    % Read the double value from Arduino
    data = fscanf(s, '%f');

    set_param(['pidcontrol' '/Constant1'],'Value',num2str(data));

    disp(['Received data from Arduino: ' num2str(data)]);
    % Do further processing with the data if needed
    y = [y data];
```

Step 7: Infinity loop

After plot y[] you will see awful thing you have done, of course it's real system not the Simulink simulate one, so adjust the data, either the PWM value or the PID slope or Arduino code, response time, threshold, anything relate to project, compare and find out the best result for it, that's why I called it infinity loop. Once you've demonstrated sufficient courage, God will grant you this outcome.

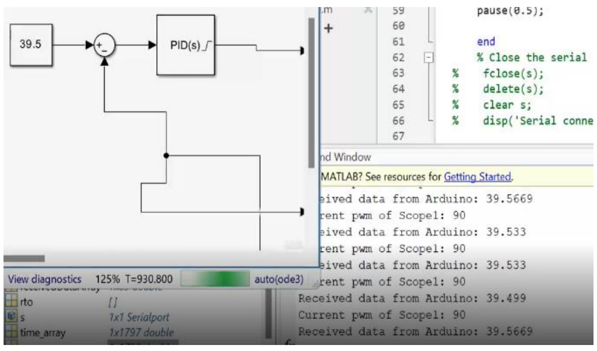


Some process:



```
    y = [y data];

    % Store sensor va
    % Wait for a mome
    pause(0.5);

    end
    % Close the seria
%    fclose(s);
%    delete(s);
%    clear s;
%    disp('Serial con
```

```
MATLAB? See resources for Getting Started.
eived data from Arduino: 35.4927
rent pwm of Scope1: 123
eived data from Arduino: 35.4927
rent pwm of Scope1: 123
eived data from Arduino: 35.5265
rent pwm of Scope1: 123
Received data from Arduino: 35.5265
Current pwm of Scope1: 123
Received data from Arduino: 35.4588
```



```
    pause(0.5);

    end
    % Close the serial
%    fclose(s);
%    delete(s);
%    clear s;
%    disp('Serial conne
```

```
MATLAB? See resources for Getting Started.
eived data from Arduino: 39.5669
rent pwm of Scope1: 90
eived data from Arduino: 39.533
rent pwm of Scope1: 90
eived data from Arduino: 39.533
rent pwm of Scope1: 90
Received data from Arduino: 39.499
Current pwm of Scope1: 90
Received data from Arduino: 39.5669
```

Although I've tried to simplify everything, this work is just so long. I hope you have gained some new knowledge from this share. Thanks for reading.

-- ShinJoy991 --