AARHUS
UNIVERSITY
AARHUS UNIVERSITY SCHOOL OF ENGINEERING

I3GFV

Experiment

Automatic Temperature Regulator

Purpose

The main purpose of these experiments is to understand the basic principles of closed loop control systems exemplified by a PID controller.

You will use a software implementation of a PID controller in the PSoC and use it to reach and maintain a constant temperature. You will also examine the effects of choosing different coefficients for the P and I parts of the controller.

The experiments should end up with a small journal and the PSoC creator projects.

Literature

- Datasheets.
- PSoC Manuals.

The relevant documents can be downloaded from BlackBoard.

General guidelines

Document the experiments in a journal.

Describe the experiment objective(s), results and reflect upon the results.

Document the test setup with photos and diagrams.

Note which components you use. Which type of motor, which sensor etc.

Document the electrical wiring and create oscilloscope/logic analyzer dumps, where you find it appropriate.

Include relevant parts of datasheets or other documentation. The relevant parts are often diagrams and illustrations.

Keep a good structure in the code and document the code.

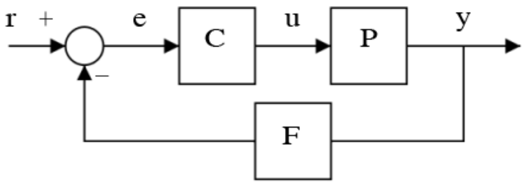
Perform the experiments in a structured manner: Think -> Do -> Document -> Reflect. And possibly iterate.

Conclude upon the results:

- What worked?
- What didn't work?
- Did anything surprise you?
- What caused the most problems?

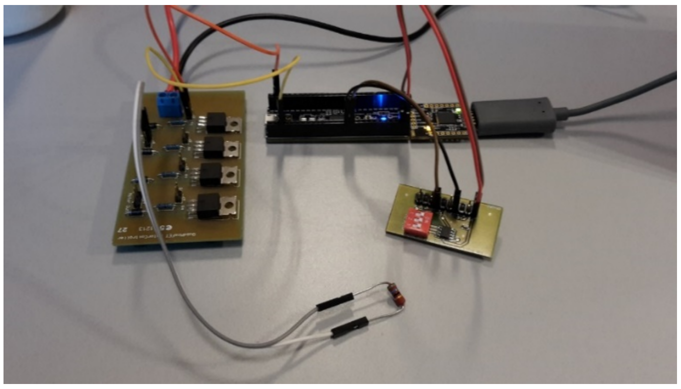
Experiment setup

The block diagram of a closed loop control system is:



Figur 1 Closed loop controller

The output of the system y is fed back through a sensor measurement F to a comparison with the reference value r. The controller C then takes the error e (difference) between the reference and the output to change the inputs u to the system under control P.

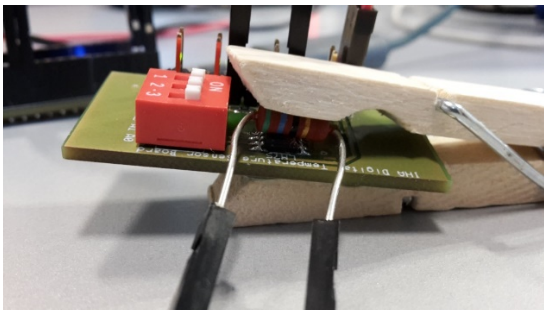


Figur 2 Photo of the system

The heater (actuator) is implemented as a 2W power resistor connected to the same MOSFET PCB you used in the motor control experiment. The controller output u is the input to the heater.

The temperature sensor is the LM75 on the same PCB you used in the communication buses experiment.

The PID controller shall be implemented in the PSoC. You can use the implementation provided on Blackboard or write your own.

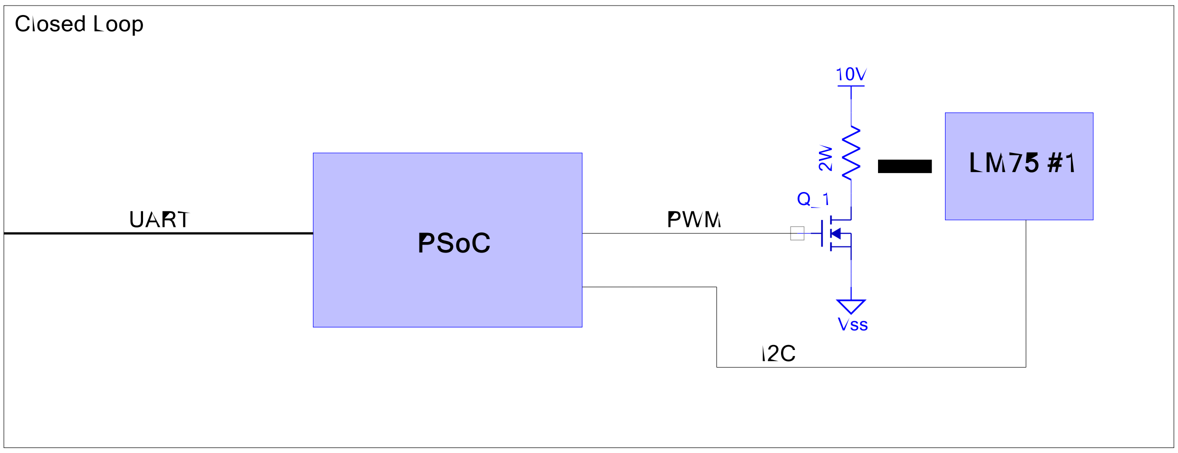


Figur 3 Connection between the power resistor and LM75

Make sure to have close contact between the power resistor and the LM75. But don't short any of the connectors on the PCB or the LM75.

AARHUS UNIVERSITY
AARHUS UNIVERSITY SCHOOL OF ENGINEERING

Experiment 1: Temperature controller



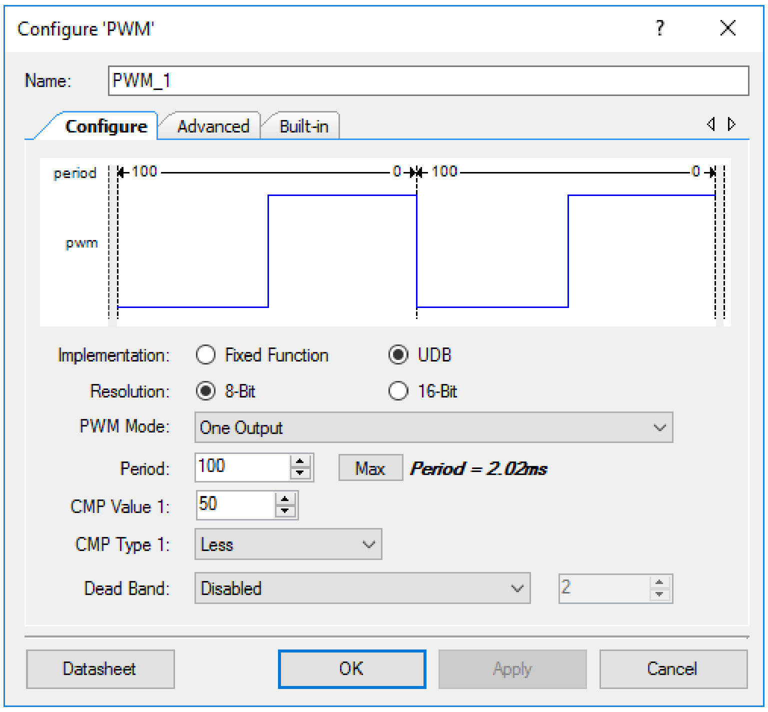
Read the temperature from the LM75 like you did in the communication busses experiment.

You can control the power dissipated in the power resistor with a PWM signal connected to the gate on a MOSFET, which powers the resistor. This is similar to how you controlled the motor speed in the motor control experiment. Use the external power supply to provide 10 Volts to the resistor.

The output of the PID controller is %PWM, i.e. 0 = no output, 100 = max effect, 50 = half effect.

An easy way to obtain this is to configure the PWM component with "Period = 100" and change the CMP value according to the PID controller output.

The figure below shows how to configure the PWM:



Name: PWM_1

Implementation: ( ) Fixed Function [✓ UDB]

Resolution: [✓ 8-Bit] ( ) 16-Bit

PWM Mode: One Output

Period: 100 *Period = 2.02ms*

CMP Value 1: 50

CMP Type 1: Less

Dead Band: Disabled 2

Make sure, that the current limiter on the power supply is turned up to max, so it won't be the power supply, which limits the power to the power resistor.

Use a software PID controller in the PSoC to obtain and maintain a constant temperature, given a setpoint.

As a starting point for the controller coefficients use:

Kp = 2, Ki = 1/30, Kd = 0

The temperatures, both measured and desired, shall be output on the UART, so they can be read on a console connected to the UART (see the *Hints* section on the next page).

NOTE! Remember to include all your raw measurements (in a text file or as a spreadsheet) in the handin. Keeping the data is also necessary, if you have to go back and re-evaluate some of your conclusions.

Always start with a running controller and a stable temperature, before making the step change in setpoint. Eg. set a temperature target to 30 degrees, wait for the temperature to be stable and then change the setpoint to 50 degrees. This way, you will have the same starting point and will be able to compare the performance of the controller with different controller coefficients.

Plot the transient response, i.e. the gradual change of output (system temperature) from initial to the desired condition (the setpoint).

Using your plot of the transient response, explain how the P and I parts of the PID controller changes over time during the transition.

How does changing the coefficients for the P and I parts affect the controller? (use a systematic approach and only change one coefficient at the time, start by changing Kp). Explain the effects using plots of the transient response with different coefficients.

What would be good PID coefficients for this temperature control system?

Does the sampling rate affect the performance of the controller?

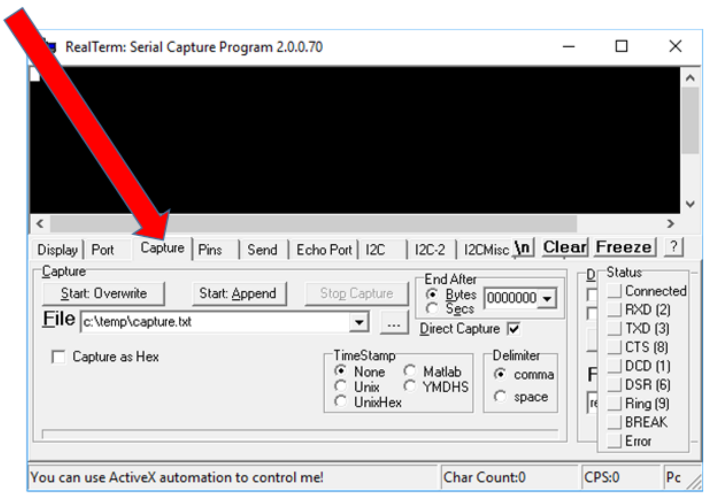
How is integral windup avoided in the controller?

Optional: What happens if you limit the integral to +-1000 ? And why?

Hints:

You may benefit from plotting the control signal and maybe even the individual outputs of the P, I and D parts of the controller, to better understand how the controller works.

If you use the provided PID controller code, you can write the console output to a file e.g. using RealTerm and import the comma separated values directly into Excel.


RealTerm: Serial Capture Program 2.0.0.70

File c:\temp\capture.txt

End After: [✓ Bytes] Secs 0000000

Direct Capture ✓

TimeStamp: [✓ None] Matlab Unix YMDHS UnixHex

Delimiter: [✓ comma] space

You can use ActiveX automation to control me! | Char Count:0 | CPS:0