TTK4175 - Instrumentation Systems
ABB

Håkon Espeland – 772703
Ali Al-Jumaili – 772170

Group 05

24. January 2017

NTNU – Trondheim
Norwegian University of
Science and Technology

Contents

**Introduction

In the course TTK4175 - Instrumentation Systems we completed our first out of a total of 5 lab assignments, the ABB lab. Throughout this assignment, basic functions of the ABB software **ControllIT** and **Control Builder Professional 4** are presented by a "Rectangle Tutorial" and a second part consisting of testing the hardware.

The purpose of the lab is to develop a software application for the process, that is; heating water to a desirable temperature by generating power to the heater from a motor and a generator using a PID-controller. The lab is built up by several parts:

Part 1 - Rectangle Tutorial

Part 2 - Testing Hardware

Part 3 - Display for control and overview of process

Part 4 - Temperature-regulator

Each part is closer described in the preceding chapters.

1 Part 1: Rectangle Tutorial

1. The Rectangle Tutorial was completed 19.01.2017, according to the documentation found on **TTK4175s** website.

2. Differences between "**Control Module**", "**Control Module Type**" and "**Single Control Module**":

Planning an automation solution, you have to choose the application layer protocol, MMS or IAC.

Control Module: Used by MMS along with function blocks. The difference between a control module and a function block, is that the control module uses automatic code sorting of code blocks, which gives better execution performance since the code blocks are executed in the most optimized way.

Control Module Type: Parent directory containing all the control modules and the single control modules.

Single Control Module: Used by Inter Application Communication along with diagrams for communication.

2 Part 2: Testing hardware

Testing the hardware was done by importing the lab template, then examination the connected IO modules. By inspection the IO parameters were identified, this gave the possibility to do the following:

- Read the status of the motor
- Connect and disconnect the load.
- Find the relationship between reference value real RPM.

2.1 Relationship between speed and reference value

The relationship was found by experimenting and reading the RPM speed from the LCD on the DCS 400 system. For each 1000 the rpm value increased by 75.2, this gives $\frac{1000}{75.2}$, by dividing the reference value by a factor of 13.3 as shown in Figure 2 we can compute the real RPM speed.

2.2 Effects of connecting the load

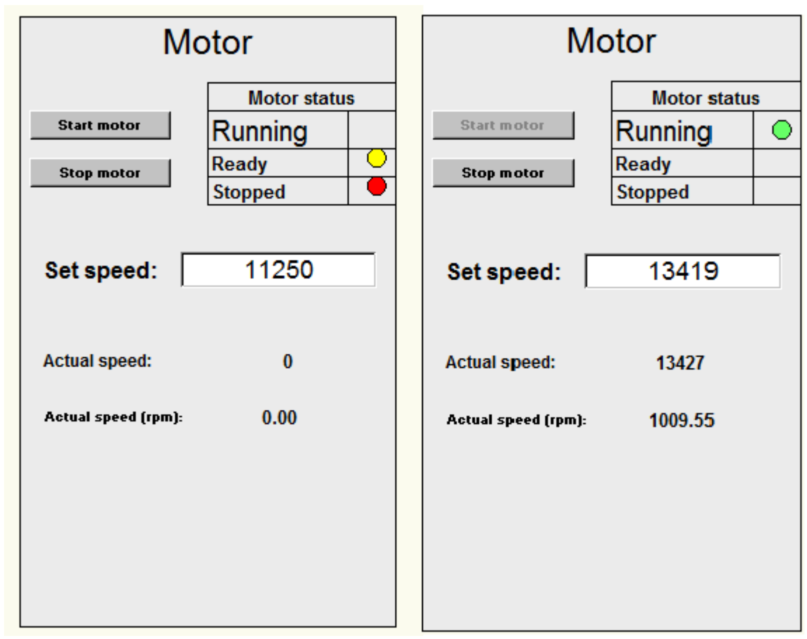
Connecting the load causes the speed to drop slightly, this is explained by that the power = rotational speed x torque. As the load is applied the motor draws more current, which increases torque. However as current flows through their windings their resistance causes the effective voltage to drop, so speed decreases

3 Part 3: Display for control and overview of the process

This section gives an overview of the modules used in the project. As shown in Figure 1 a library was created containing control modules for the Generator, Heater and Motor

3.1 Motor Faceplate

From Figure 1 information about the motor status, running (green), stopped (red), ready (yellow) is given. The speed can be manually controlled. As well to reading the actual output from the IO modules, and the calculated RPM value.



Set speed: 11250

Actual speed: 0

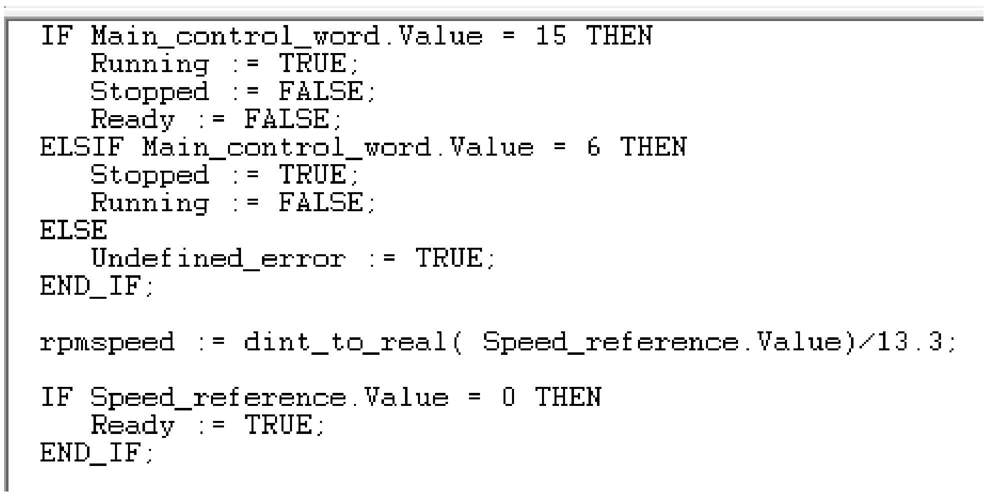
Actual speed (rpm): 0.00

Set speed: 13419

Actual speed: 13427

Actual speed (rpm): 1009.55

Figure 1: Motor Faceplate



```
IF Main_control_word.Value = 15 THEN
   Running := TRUE;
   Stopped := FALSE;
   Ready := FALSE;
ELSIF Main_control_word.Value = 6 THEN
   Stopped := TRUE;
   Running := FALSE;
ELSE
   Undefined_error := TRUE;
END_IF;

rpmspeed := dint_to_real( Speed_reference.Value)/13.3;

IF Speed_reference.Value = 0 THEN
   Ready := TRUE;
END_IF;
```

Figure 2: Motor Structured Text

3.2 Generator Faceplate

Figure 3 shows the faceplate for the generator, this gives information about the power consumption of the load, as well to connecting/disconnecting the load.

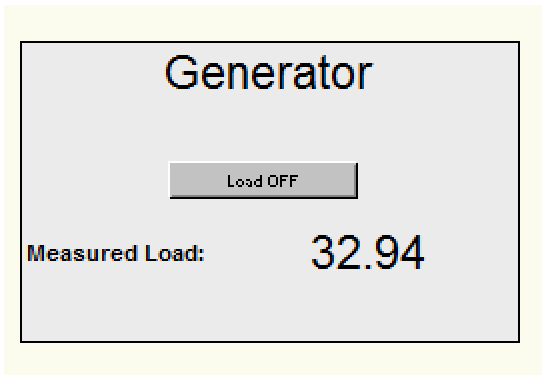


Figure 3: Generator Faceplate

3.3 Heater Faceplate

Figure 4 shows the heaters faceplate, which gives the the current temperature value (under the slope), the states of both pressure and temperature (on/off).

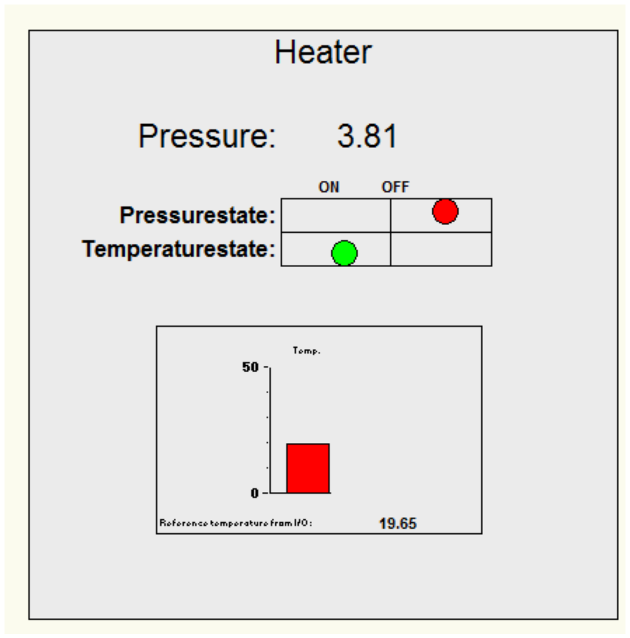
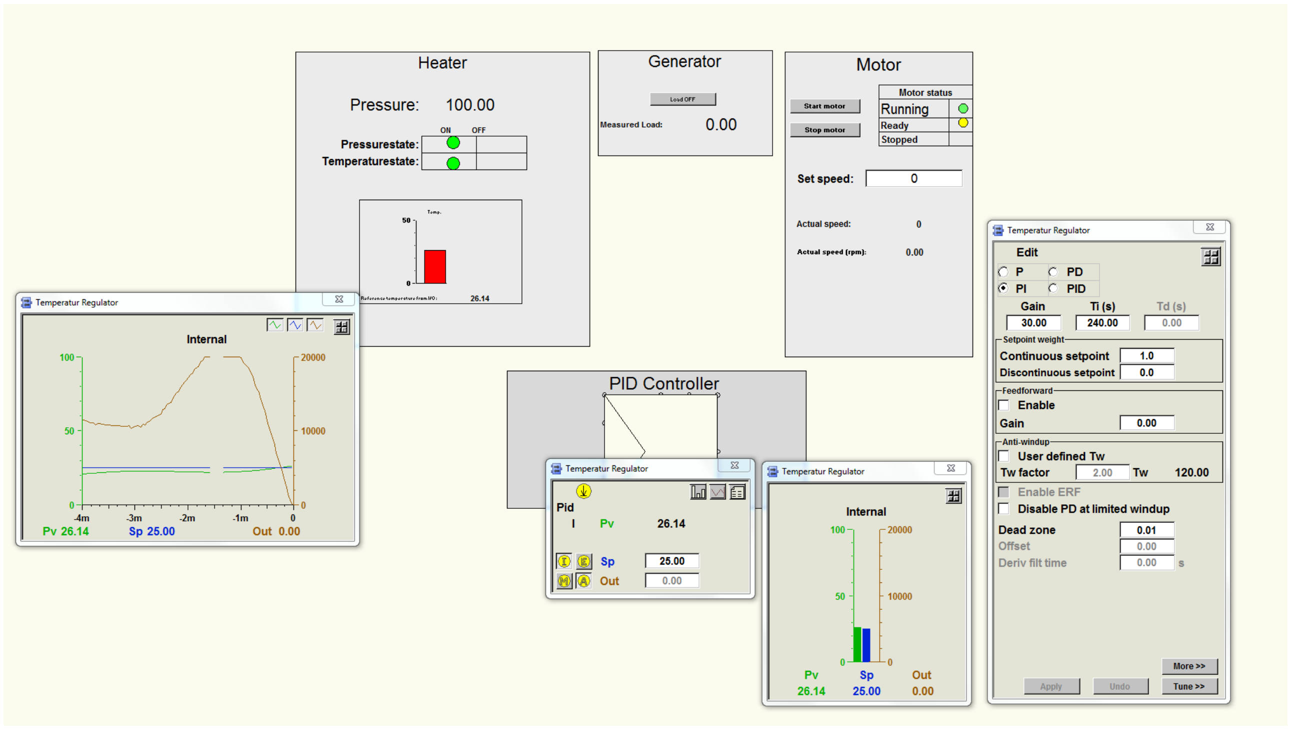


Figure 4: Heater Faceplate

3.4 Overview of the Process

A human interface of the motor, generator, heater and temperature regulator is shown in Figure 5. While Figure 6 shows the structure of the program.



Pressure: 100.00

	ON	OFF
Pressurestate:	●	
Temperaturestate:	●	

Measured Load: 0.00

Motor status	
Running	●
Ready	●
Stopped	

Set speed: 0

Actual speed: 0

Actual speed (rpm): 0.00

Pv 26.14 Sp 25.00 Out 0.00

Pid

I Pv 26.14

Sp 25.00

Out 0.00

( ) P ( ) PD

(•) PI ( ) PID

Gain	Ti (s)	Td (s)
30.00	240.00	0.00

Setpoint weight

Continuous setpoint	1.0
Discontinuous setpoint	0.0

Feedforward

[ ] Enable

Gain 0.00

Anti-windup

[ ] User defined Tw

Tw factor 2.00 Tw 120.00

[ ] Enable ERF

[ ] Disable PD at limited windup

Dead zone	0.01	
Offset	0.00	
Deriv filt time	0.00	s

Figure 5: Overview HMI

4 Part 4: Temperature-regulator

4.1 Developing the controller

This can be done by adding a PID controller block, then connecting the input signal to the set value of the wanted temperature, and the output to the speed of the motor, the generator produces energy which is transferred to the heating element inside of the water tank, heating up the water.

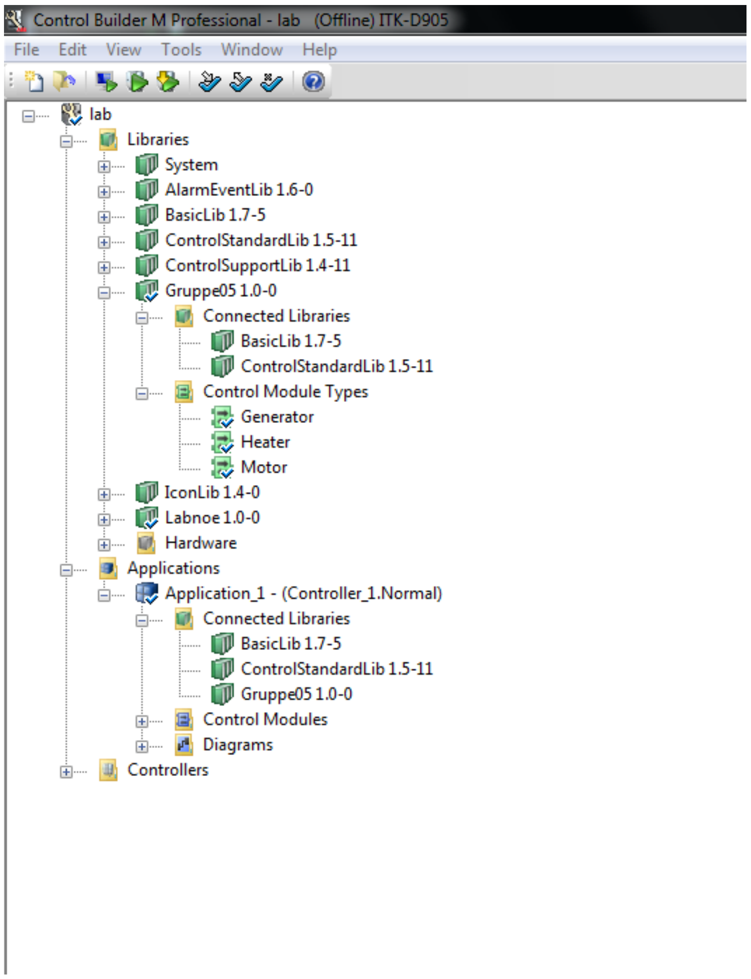
After completing Part 1-3, we started developing a temperature-regulator based on the applied load. As mentioned in the assignment text, we chose to implement the PidCC from the Standard Library, using four of its connections:

tempset → Real_to_CC → Input SV @ PID.

M4_PT100.Value → Real_to_CC → Input PV @ PID.

Ouput_CC @ PID → CC_to_int → Output to Speed_Setspeed.Value

And we also changed the maximum range to the parameter Maxspeed which is equal to 20 000.


Control Builder M Professional - lab (Offline) ITK-D905
File Edit View Tools Window Help

```
lab
  Libraries
    System
    AlarmEventLib 1.6-0
    BasicLib 1.7-5
    ControlStandardLib 1.5-11
    ControlSupportLib 1.4-11
    Gruppe05 1.0-0
      Connected Libraries
        BasicLib 1.7-5
        ControlStandardLib 1.5-11
      Control Module Types
        Generator
        Heater
        Motor
    IconLib 1.4-0
    Labnoe 1.0-0
    Hardware
  Applications
    Application_1 - (Controller_1.Normal)
      Connected Libraries
        BasicLib 1.7-5
        ControlStandardLib 1.5-11
        Gruppe05 1.0-0
      Control Modules
      Diagrams
  Controllers
```

Figure 6: Overview structure

One of the **challenges** we faced during this part was to create the controller as a control module type. This caused a lot of errors, and made us complete this part inside the CDM editor of the application. Then the connections were a lot easier to implement, and by using the "invisible" function of the RealToCC and CCToInteger blocks, we were able to make a smooth interaction object for the PID.

4.2 Tuning the PID Controller

The tuning started out with the PID-parameters provided in the assignment text.

$\rightarrow$ **P** = 3.0

$\rightarrow$ **I** = 240 (seconds)

$\rightarrow$ **dead zone** = 0.01 (Removing noise)

$\rightarrow$ **direction** = REVERSE

These parameters provided a rather slow response. Then we disabled both the **I** and the **D**, and increased the **P** until we found a suitable gain, befor we enabled the **I** back to 240 seconds again.

Water and fluids in general has relatively high heat capacity, and responds very slow to changes in temperature. But in the end, with a **gain** of **30** and **I = 240**, the PID controller anticipated the heat capacity, and decreased the output long before the set-value was reached. This resulted in a regulation of +/**- 0.5** degrees celsius.

Some thoughts around the regulation is to decrease the maximum range of the output if this has not already been accounted for. If the maximum speed of the motor really was 20 000, it would not be ideal to make the PID-regulator output driving the motor at full speed if the resulting temperature would be almost as good if the maximum range was set to 18 000 or lower.

5 Conclusion

Completing the ABB lab has given us a general introduction to the tools used in conjuction with the ABB 800xA Control System. All sections were completed with focus on HSE, with real-life safety-implementations, as for example the motor: It is not possible to start before the motor is totally stopped, when the speed is 0. Indicators as flashing lights for pressurestate, temperaturestate, running, ready and stopped were also implemented in the application.

The PID controller could probably be tuned more precisely, but functioned in a satisfying way for this specific lab.