Real Time System Architectures
Assignment 1

Ahmed Semih Özmekik
171044039

March 18, 2021

Abstract

Shows a diagram describing the system in the design of the controller, with the code for loop, control, read and write operations..

1 Objective

Design a controller for to keep gas temperature and pressure constant.

2 Design

2.1 Overview

There are 3 different flows in the system that we can consider as threads and work flows. First of all, let's get to know them.

There are two similar jobs that have almost the same workflows, but different hertz values for reading via ADC: Temperature and Pressure. These streams set their own processes independently from each other, so each stream works to keep its value above a certain threshold value. These threads work asynchronously.

The third thread is a process developed for Display. This is designed to read and print values from the other two threads. Thus, this display process must work synchronously with the other two processes.

Let's show the system, which we can examine the general flow in this way, with both diagram and pseudocode.

2.2 Synchronization

With these processes, the display needs to work synchronously with the pressure and temperature values. In order to achieve this, it is necessary to plan a draft.

The first process (temperature) works at 100 Hz. The second process (pressure) works at 10 Hz. If the display process also runs at 100Hz, that is, it adapts to the faster worker, these three processes will work synchronously and the values will always remain up to date.

3 Implementation

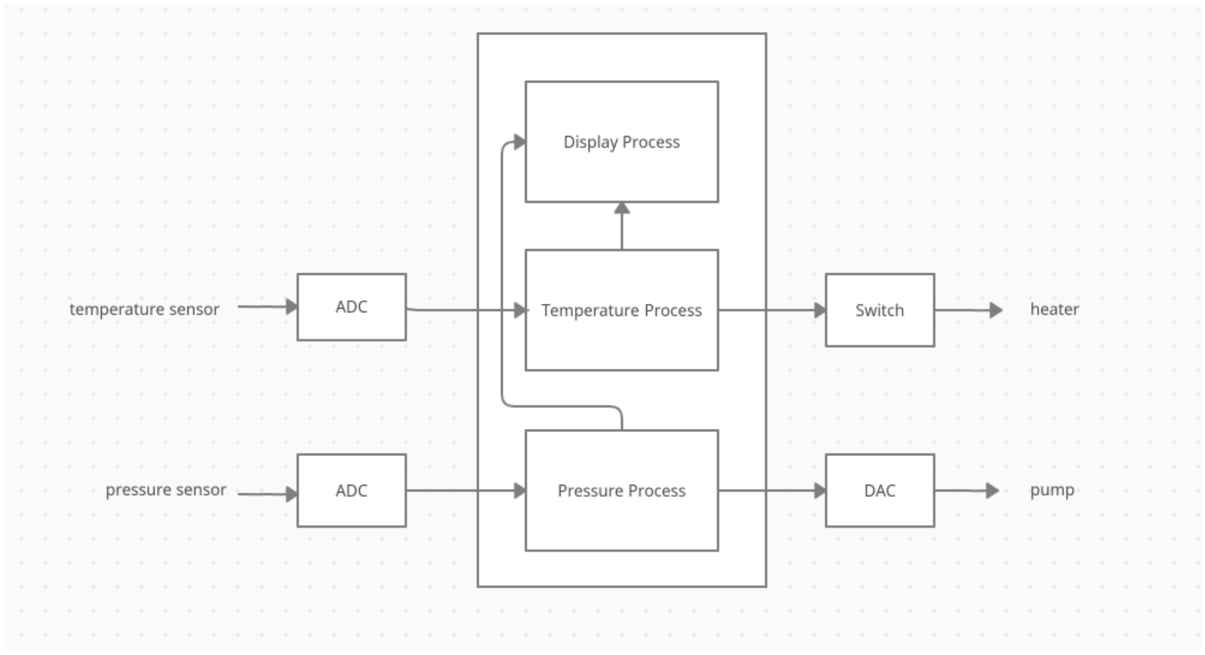

Figure 1: Block Diagram

3.1 Pressure

```
while (True) {
    t0 = time();

    adc_trigger(pressure_port);

    p = read_adc(pressure_port);

    /* routine for to control pressure p value */
    check_pressure(p);

    write_dac(p);

    t1 = time();

    /* 100 Hz */
    sleep(10 - (t1 - t0));
}
```

3.2 Temperature

```
while (True) {
    t0 = time();

    adc_trigger(temp_port);

    t = read_adc(temp_port);

    /* subroutine for to control temperature t value */
    check_temp(t);

    write_switch(t);

    t1 = time();

    /* 10 Hz */
    sleep(100 - (t1 - t0));
}
```

3.3 Display

```
while (True) {
    t0 = time();

    /* subroutine for to display values */
    print();

    t1 = time();

    /* 100 Hz */
    sleep(10 - (t1 - t0));
}
```