TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

Report of laboratory Work №5

Control

Student: Pogorevici Daniel, FAF-202
Teacher: univ.lecturer Moraru Dumitru

Chisinau 2023

1. The task of the laboratory work:

2. Develop an MCU-based application that will implement management systems for

a) regulation of temperature or humidity with the application of the On-Off driving method with hysteresis with relay operation

b) engine speed adjustment with the application of the PID method with an encoder as a sensor, and L298 driver for the application of power to the engine.

NOTE: in p (b) you can choose another control parameter, with the constraint that the drive will have a resolution of at least 8 bits.

3. The set point (control setpoint) will be set from one of the sources of your choice

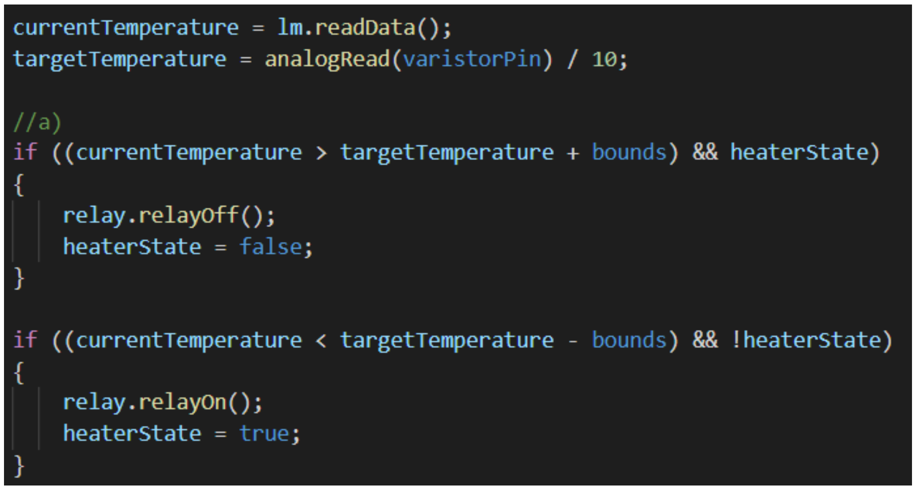
- a potentiometer
- two buttons for UP / Down
- encoder sensor
- keypad
- serial interface

4. The Setpoint and Current values will be displayed on the LCD

2. The progress of the work

To realize this issue I have used a heater, LM35 analog temperature sensor and varistor(to set target temperature).

Appendix 1



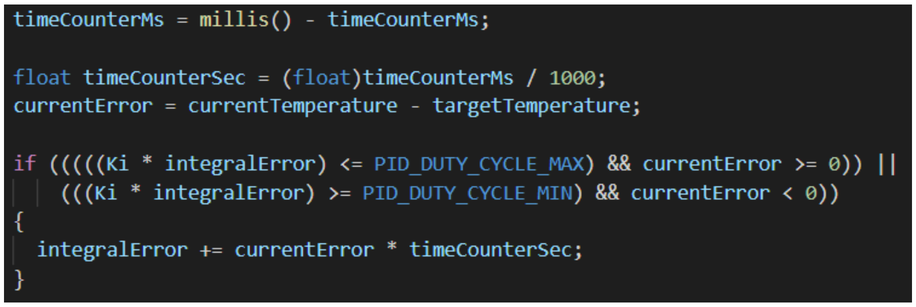
```
currentTemperature = lm.readData();
targetTemperature = analogRead(varistorPin) / 10;

//a)
if ((currentTemperature > targetTemperature + bounds) && heaterState)
{
    relay.relayOff();
    heaterState = false;
}

if ((currentTemperature < targetTemperature - bounds) && !heaterState)
{
    relay.relayOn();
    heaterState = true;
}
```

Fig 1. regulation of temperature driving method with hysteresis with relay operation

We read data from the sensor and compare it with the target + or - bounds. If it crosses the border then we turn on the heater or turn off and start the PID regulator.

The task of the PID controller is to ensure that the temperature of the sensor is equal to the target value, which will be set programmatically. Direct adjustment will occur by changing the fan speed, by changing the duty cycle of the PWM signal.

Appendix 2



```
timeCounterMs = millis() - timeCounterMs;

float timeCounterSec = (float)timeCounterMs / 1000;
currentError = currentTemperature - targetTemperature;

if (((((Ki * integralError) <= PID_DUTY_CYCLE_MAX) && currentError >= 0)) ||
    (((Ki * integralError) >= PID_DUTY_CYCLE_MIN) && currentError < 0))
{
  integralError += currentError * timeCounterSec;
}
```

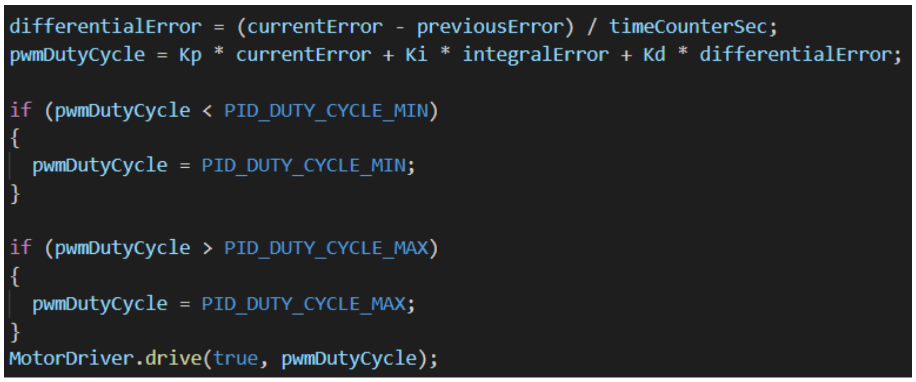
Fig 2. An additional constraint for the integrating component

I introduced an additional constraint for the integrating component here so that too large values do not accumulate, which can lead to increased system inertia(Appendix 2).

If the current error is positive, and at the same time the value of the integrating component does not exceed the maximum value of the PWM signal pulse duration, then we accumulate the error in *integralError*, otherwise we do not. The same is true for a negative error value.

Next, we differentiate the residual and calculate the output.

Appendix 3



```
differentialError = (currentError - previousError) / timeCounterSec;
pwmDutyCycle = Kp * currentError + Ki * integralError + Kd * differentialError;

if (pwmDutyCycle < PID_DUTY_CYCLE_MIN)
{
  pwmDutyCycle = PID_DUTY_CYCLE_MIN;
}

if (pwmDutyCycle > PID_DUTY_CYCLE_MAX)
{
  pwmDutyCycle = PID_DUTY_CYCLE_MAX;
}
MotorDriver.drive(true, pwmDutyCycle);
```

Fig 3. Calculating output

Also we need to check if *pwmDutyCycle* doesn't exceed bounds.

The meaning of using a PID controller is that it will provide control of the parameter regardless of changes in external uncontrolled factors. Mathematically, the principle of operation can be represented very simply: $y(t)=f(e(t))$

Where:

- $y(t)$ - input signal
- $e(t)$ - difference between the target and the current value of the controlled variable.

And the PID controller gives us a mechanism to calculate *y(t)* from *e(t)*. Again I refer to a future example in which everything will finally fall into place. In the meantime, we are systematically moving to consider how exactly these calculations occur. The controller output signal is defined as follows:

$$y(t) = P + I + D = K_p \times e(t) + K_i \times \int_0^t ■ e(\tau)d\tau + K_e \times \frac{de}{dt}$$

We have the algebraic sum of three components, which gave the name to the controller - PID:

- $K_p \times e(t)$ - proportional component
- $K_i \times \int_0^t ■ e(\tau)d\tau$ - integrating component
- $K_e \times \frac{de}{dt}$ - differential component

It should be noted right away that not all components can be used, but only a part of them, then the regulator will be called proportionally differentiating, proportionally integrating, etc. The logic of forming names here is simple and obvious.

The formula has three uncertain values, the selection of which is the setting of the PID controller. We are talking about the gains of the proportional, integrating and differentiating components (K_p, K_i , K_e) that we can calculate using different methods, for example Ziegler-Nichols method.

The Ziegler-Nichols method in the sequential execution of the following operations:

1. To reset all coefficients of the regulator.
2. To set some target value of the controlled parameter (for example, temperature).
3. To begin to increase gradually the proportional coefficient and monitor the reaction of the system.
4. At a certain value of K_p, undamped oscillations of the controlled variable will occur.
5. To fix this value, as well as the oscillation period of the system.

This concludes the practical part of the method. From the obtained values we calculate the coefficients:

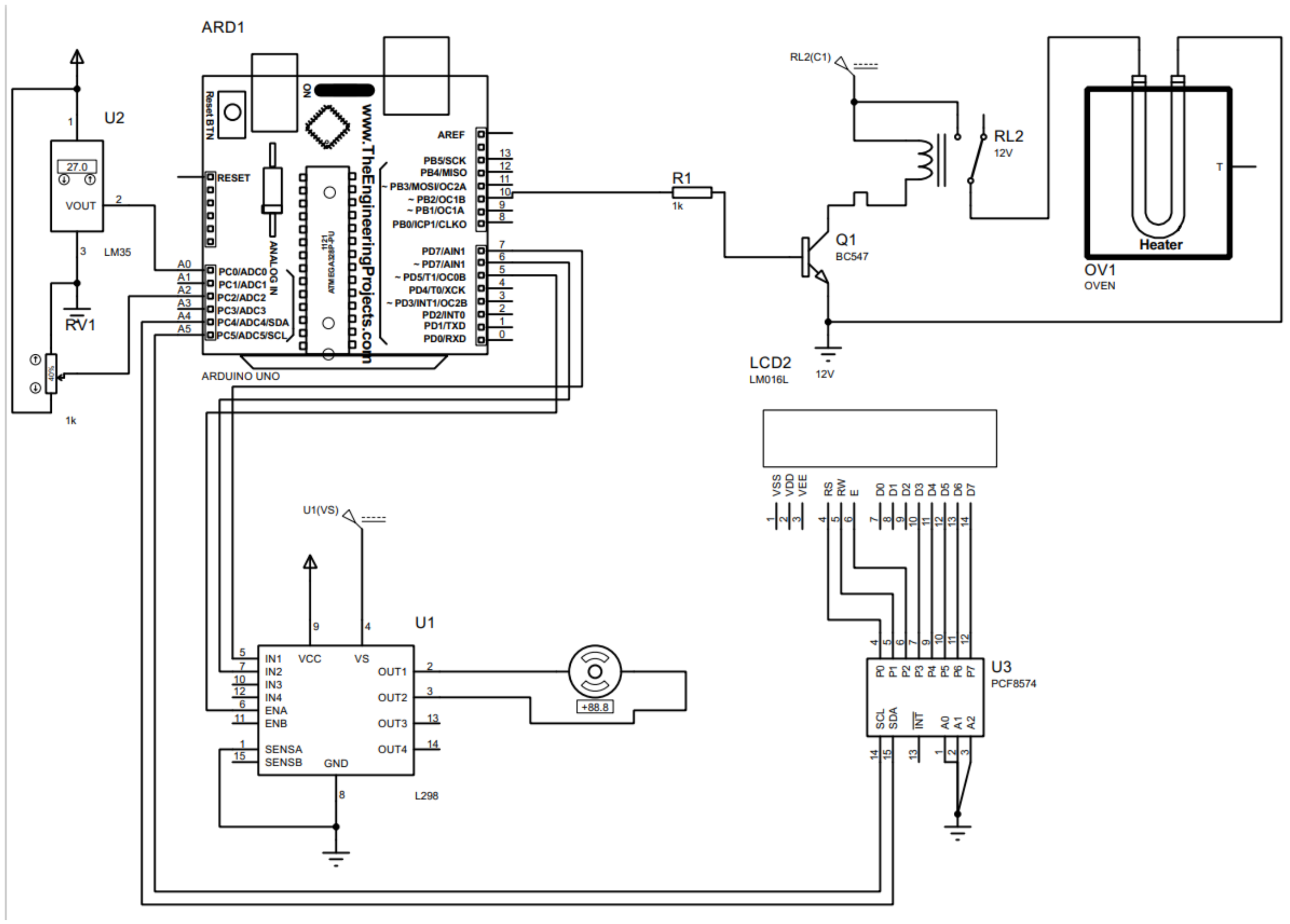
$$K_p = 0.6 \times K$$

$$K_i = (2 \times K_p) / T$$

$$K_d = (K_p \times T) / 8$$

Here *K* is the same coefficient of the proportional component at which oscillations occurred, and *T* is the period of these oscillations.

3. Electrical scheme in proteus



4. Conclusion

After performing the 5th laboratory work, I have learned a lot of new information about PID controllers, have seen examples of their realizations on other platforms.

In the first task, the On-Off driving method with hysteresis and relay operation is applied to regulate the temperature or humidity. In the second task, the PID method with an encoder as a sensor and L298 driver for the application of power to the engine is implemented for engine speed adjustment. The control setpoint can be set from various sources such as a potentiometer, buttons for UP/Down, encoder sensor, keypad, or serial interface. The Setpoint and Current values are displayed on the LCD and Terminal.

Overall, this laboratory work is a good exercise for learning about MCU-based applications and control systems. It provides an opportunity to apply various control methods and sensors, and also to display the results on an LCD screen and Terminal.