**TCES445 – DC Water Tank Level Control**

Final Project Report

Group Members:

*James Brewer*

*Igor Gonchar*

Dec 11, 2015

Table of Contents:

Project Objective: 2

Description of System: 2

Pre-experimental data: 3

Theory of Operation: 6

Software Simulation: 7

Hardware: 9

Implementation 9

Conclusion: 11

Appendix: 12

List of Figures:

Figure 1: *Closed Loop System* 2

Figure 2: Ideal tank behavior 6

Figure 3: *Tank transfer function* 7

Figure 4: The complete block diagram of the implementation 9

List of Tables:

Table 1: *This table represents our measurements for the fluid resistance of water.* 3

Table 2: *This table represents our calculations for fluid resistance* 4

Table 3: *This table represents the time to fill the tank with 200mL of water.* 5

Table 4: *This table represents our data for the pump constant.* 5

Table 5: *This table represents the eTape measurements.* 6

List of Graphs:

Graph 1: *This graph represents the trendline and equation for our eTape.* 6

Graph 3: *This graph represents the step response of our system with a controller* 8

Graph 2: *This plot represents the step response of our system without a controller.* 8

# Project Objective:

The objective of this report was to develop a feedback control system that regulates the water level in a tank. Our target was to produce a PID controller via the Arduino platform that could regulate the water level in a small tank. A secondary objective was to further analyze and gain experience in real-world feedback systems.

# Description of System:

*As taken from our project instructions:*

“During normal operation, the valve at the bottom of the tank is open and the control system controls the speed of the pump, which, in turn, controls the rate of flow into the tank. The tank level is measured by a pressure sensitive tape, which changes resistance as the water level changes. A potentiometer allows the user to change the water level set point”.

The physical board that was used for this project was the Arduino Mega2560

*A more visual description of our system is shown below in Figure 1.*

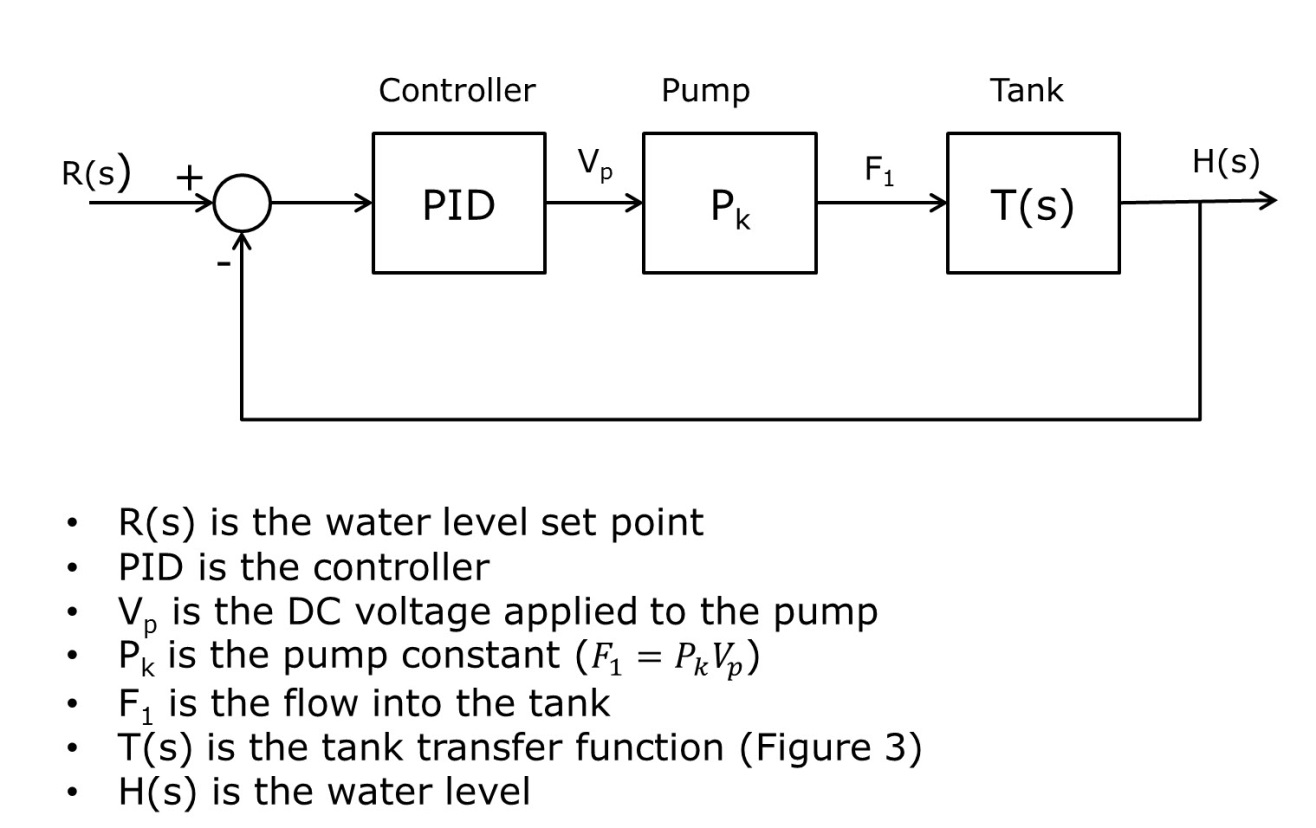


Figure 1: *Closed Loop System*

# Pre-experimental data:

*This section underlines our measurement process and tests to find any necessary constants for the system’s transfer function.*

First we measured the constant tank cross sectional area (A). This measurement required simply measuring the diameter of the tank opening, and converting it into area.

Diameter of the tank: 3.5cm

Then we began measuring the fluid resistance of our liquid in the tank (water). In order to measure the fluid resistance of water, we ran multiple tests measuring the time it took to drain a specific amount of water from the tank. These tests are shown below in the following table:

Table 1: *This table represents our measurements for the fluid resistance of water.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Volume (mL) | Height (cm) | Trial #1  (sec) | Trial #2  (sec) | Average (sec) | Flow rate (mL/sec) |
| 100 | 4.2 | 3.75 | 3.32 | **3.54** | **28.25** |
| 150 | 10.4 | 5.41 | 5.23 | **5.32** | **28.20** |
| 200 | 15.3 | 6.11 | 5.76 | **5.94** | **33.67** |
| 250 | 21.0 | 7.24 | 7.17 | **7.21** | **34.67** |
| 300 | 25.8 | 7.72 | 7.86 | **7.79** | **38.51** |

This table demonstrates the time that it took for the water to drain out of the tank. As you can see, we measured our times with varying water volumes (consequently varying heights). The reason why these additional measurements are necessary is because the flow rate out of a tank varies with height of water in the tank. We wanted to average the times from multiple heights, to take this pressure difference into account.

To calculate fluid resistance, we were given the assumption that the height of the water in the tank is directly proportional to the flow rate. Therefore,

*The following table represents our calculations for the fluid resistance.*

Table 2: *This table represents our calculations for fluid resistance*

|  |  |  |  |
| --- | --- | --- | --- |
| Flow rate (mL/sec) | Height (cm) | 1/R | R |
| 28.25 | 10.2 | 2.77 | 0.361 |
| 28.20 | 16.4 | 1.72 | 0.582 |
| 33.67 | 21.3 | 1.58 | 0.633 |
| 34.67 | 27.0 | 1.28 | 0.779 |
| 38.51 | 31.8 | 1.21 | 0.826 |
| Average | | **1.712** | **0.584** |

We found the coefficient of fluid resistance of water to be:

The final pre-experimental measurement required was the water pump constant (). We will go into further detail about the water pump specifications in a later section.

To calculate the water pump constant, we assumed that the flow of water was directly proportional to product of the voltage of the pump and the pump constant.

To find the flow rate, we divided a constant volume of water (200mL) by the average time it took to fill the tank. The following table represents our findings.

Table 3: *This table represents the time to fill the tank with 200mL of water.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pump Voltage (v) | Trial #1 (sec) | Trial #2 (sec) | Average (sec) | Flow rate (mL/sec) |
| 3.294 | 7.14 | 7.16 | 7.15 | 27.97 |
| 6.165 | 4.64 | 4.58 | 4.61 | 43.38 |
| 9.082 | 3.62 | 3.76 | 3.69 | 54.20 |
| 12 | 3.03 | 3.19 | 3.11 | 64.31 |

Under experimentation, the behavior is not linear; therefore we conducted multiple trails and averaged the pump constant from them all. The following table represents our calculations for the water pump constant.

Table 4: *This table represents our data for the pump constant.*

|  |  |  |
| --- | --- | --- |
| Pump Voltage (v) | Flow rate (mL/sec) | Pump constant |
| 3.294 | 27.97 | 8.49 |
| 6.165 | 43.38 | 7.04 |
| 9.082 | 54.20 | 5.97 |
| 12 | 64.31 | 5.35 |
| Average: | | 6.7125 |

Therefore, we found the water-pump constant to be:

Finally, we had to measure the resistance of the eTape and mapped the height of the water to the ADC (Analog to Digital Converter) values of the voltage divider circuit. This allowed us to create a scatter plot of the data, and via EXCEL Software, to retrieve the trend line equation. The table and graph can be seen below.

Table 5: *This table represents the eTape measurements.*

|  |  |
| --- | --- |
| ADC | Height (cm) |
| 522 | 6 |
| 505 | 8.2 |
| 479 | 10.9 |
| 450 | 13.3 |
| 443 | 14.2 |
| 438 | 14.6 |
| 430 | 15.1 |
| 400 | 17.4 |
| 367 | 19.8 |
| 339 | 22.1 |
| 309 | 24.5 |

Graph : *This graph represents the trendline and equation for our eTape.*

We used MATLAB via the command window, and found more significant figures for the coefficients.

# Theory of Operation:

In theory, the water tank has ideal properties (meaning that no water is lost, or that the outflow rate of water is directly proportional to the product of the height and the reciprocal of the fluid resistance).

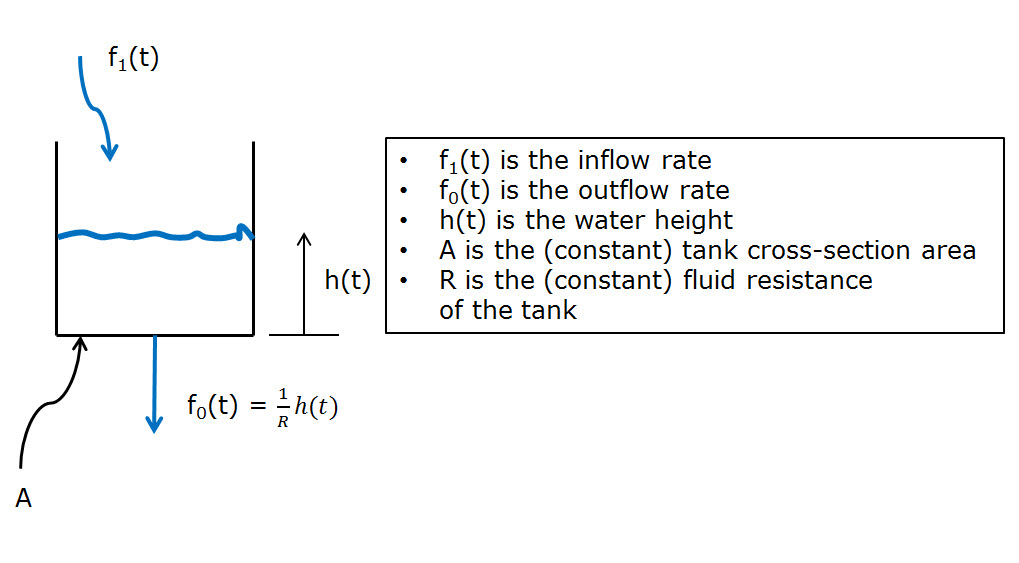
The following figure demonstrates the assumed ideal properties.

Figure : Ideal tank behavior

We were given the governing differential equation for the inflow rate to be:

We were also given the transfer function as shown below:

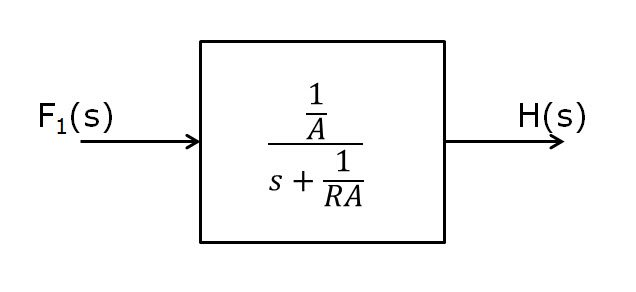


Figure 3: *Tank transfer function*

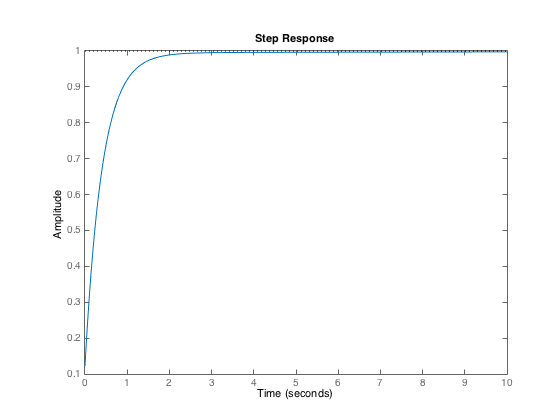
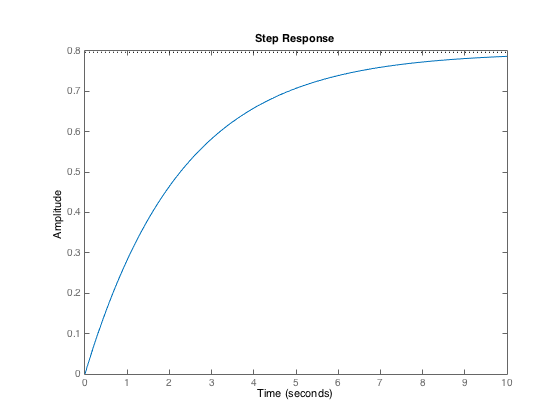
From this information, we were able to implement an Arduino microcontroller to take in an ADC value from both the potentiometer and the eTape. Converting those to height values, we computed our error and gave that value to an algorithm, implementing our PID controller digitally. The PID controller outputs a voltage value for the water pump, which we converted to a PWM value (0-255), to control the flow rate and set the water level, keeping it stable.

# Software Simulation:

To achieve a quick rise time, minimal overshoot, and a correct water level, we had to implement a PID controller. In doing so, we decided to refer to the manual tuning method in designing our controller. Below is a block diagram of our system.

We used the *Simulink* code tools and code to simulate the step response of our system. However, we opted out of using the *Block Diagram Creator*, due to personal preference.

To better visualize the effect of our controller, we plotted the step response of our system in MATLAB. Below are the plots of our original system (without a controller), compared to our final system (with an active controller).

****

Graph : *This graph represents the step response of our system with a controller*

Graph : *This plot represents the step response of our system without a controller.*

As seen above, the original system was over-damped and the steady-state value was at 0.8, meaning that it undershot. Whereas, by adding the controller, our system improved to a much quicker rise time, and reduced the error. Our steady-state value was a 1.0, as expected.

Our Original PID values are:

Experimenting with different values, we noticed that the proportional parameter decreased the rise time. The proportional parameter got us closer to our desired steady state value, but we still suffered either from undershoot or overshoot.

The integral parameter decreased the rise time and the steady-state error, however too much of this parameter causes overshoot. And finally, the deferential parameter slightly shifts the graph to the left, decreasing the rise time.

Knowing the effects that these parameters play on our system, we were able to derive appropriate values for these parameters as shown above.

# Hardware:

Water Tank/Hoses

High Flow Gold Series – Water Pump

Pololu - VNH5019 Motor Driver Carrier

MILONE 12-inch eTape (PN-12110215TC-12)

Potentiometer

Arduino Mega 2560

DC Power Supply CSI3003X III

1.8kΩ resistor + Breadboard

# Implementation

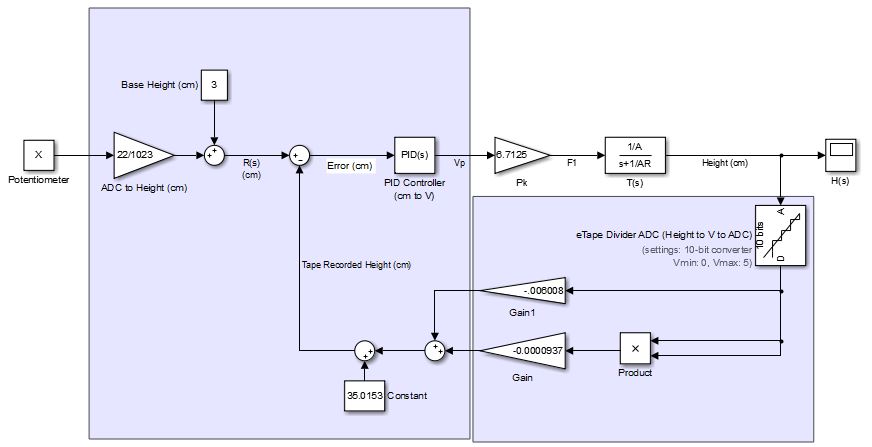


Figure 4: The complete block diagram of the implementation

When it came time to experiment with our PID controller, we set up an Arduino sketch to run the system and to display the desired height versus the actual tape height to the serial monitor. We found that more often than not, the eTape reported false heights, as it was probably an inaccurate sensor. We relied to visual inspection of the water level in the tank to confirm the actual height. During the experimental process, we did end up replacing the eTape and recalculated our pre-experimental measurements.

With our original controller parameters, we found the water height seamed to follow the desired height, however, there was some minor oscillation. In order to stabilize the system, we retested some pre-experimental values to get better averages.

After retesting and tweaking our controller parameters to the ones shown below, we found that our system remained stable for a range from ~6 cm to ~12 cm on the eTape.

Our final PID values are:

The reported values on the serial console were still unstable and inaccurate compared to the actual height on the tape itself. This means that any data recordings would have been misrepresentative of the actual system.

However, these issues did not affect our system stability since we calculated our governing equations with these issues in consideration.

# Conclusion:

We found that several components of the system were not linear (as we were told to assume), which in turn, affected the stability and accuracy of the system outside of a certain range. However, because we took averages in our pre-experimental data across different trials and settings, our system was more stabile within a median range.

Overall, we successfully implemented a closed-loop, feedback control system to monitor and stabilize the water level in a tank. We gained experience with designing a PID controller digitally, and with reflecting the transfer functions mathematically.

# Appendix:

**waterLevelControler.ino**

//TCES 455 Autumn 2015

//Lab 7

//James Brewer, Igor Gonchar

double ptmVal = 0;

double tapeVal = 0;

double tape\_vha = -0.0000937; //ADC voltage to height conversion

double tape\_vhb = -0.006008; //ADC to height intercept (-11.637x + 578)

double tape\_vhc = 35.0153;

double pot\_vh = (double)22/(double)1023;

int pot\_vhi = 3;

double error = 0; //Height error value

double last\_error = 0;

double u; //Controller output (Voltage)

double P = 0;

double I = 0;

double D = 0;

double Kp = 6;

double Ki = 0.7;

double Kd = 0.2;

int powSupply = 12; //Voltage of the power supply

double Kpwm = 255/powSupply; //Conversion factor, voltage

//to PWM value

double Vp = 0; //Pump voltage

int sampRate = 50; //Sampling rate (Hz)

double T = (double)1/(double)sampRate; //Sampling time / dt

int pinPtm = A0; //Analog pin A0 (Potentiometer)

int pinTape = A2; //Analog A2 (Tape)

int pinA = 4;

int pinB = 8;

int pinPWM = 6;

void setup() {

Serial.begin(19200);

pinMode(pinPWM, OUTPUT);

pinMode(pinA, OUTPUT);

pinMode(pinB, OUTPUT);

digitalWrite(pinA, HIGH);

digitalWrite(pinB, LOW);

noInterrupts();

TCCR3A = 0; // clear the timer-counter control registers A and B

TCCR3B = 0;

TCNT3 = 0; // clear the actual counter

TCCR3B |= (1 << WGM32); // this sets CTC mode

TCCR3B |= (1 << CS31); // 64 prescaler (CS11 and CS10)

TCCR3B |= (1 << CS30); //

TIMSK3 |= (1 << OCIE3A); // this enables the timer compare interrupt

OCR3A = (250000/sampRate) - 1; //(Potentiometer/Tape reading)

interrupts();

}

void loop() {

delay(1000/sampRate);

Serial.print(ptmVal);

Serial.print(" , ");

Serial.println(tapeVal);

//Save error values

error = ptmVal - tapeVal;

if (abs(error) < 4){ // prevent integral 'windup'

I = I + error; // accumulate the error integral

} else {

I = 0; // zero it if out of bounds

}

P = error\*Kp;

//Serial.print(P);

//Serial.print(" , ");

I = (I + (error\*T))\*Ki;

//Serial.print(I);

//Serial.print(" , ");

D = (last\_error - error)\*Kd/T;

//Serial.print(D);

//Serial.print(" , ");

Vp = P + I + D;

Vp = Vp\*Kpwm;

//Serial.println(Vp);

if (Vp > 255) {

Vp = 255;

} else if (Vp < 0) {

Vp = 0;

}

//Serial.print(Vp);

analogWrite(pinPWM, Vp);

last\_error = error;

}

ISR(TIMER3\_COMPA\_vect) // timer compare interrupt service routine

{

ptmVal = (analogRead(pinPtm)\*pot\_vh) + pot\_vhi;

tapeVal = (double)analogRead(pinTape);

tapeVal = (tapeVal\*tapeVal\*tape\_vha) + tape\_vhb\*tapeVal + tape\_vhc;

}

**lab7quickdata.m**

%

% Tank function

%

R = 0.584;

A = 19.242;

Ts = tf(1/A,[1 1/R/A]);

%

% Pump constant

%

Pk = 6.7125;

Ps = tf(Pk,1);

Gs = series(Ps, Ts);

%stepplot(Ts); %Step response of water height alone (T(s))

%stepplot(Gs); %Step response w/ pump constant

kp = 6;

ki = 0.7;

kd = 0.2;

%ti = 0.1;

%td = 0.2;

Cp = tf(kp,1); %Proportional

Ci = tf(ki, [1 0]); %Integrator

Cd = tf([kd 0],1); %Derivation

PID = parallel(Cp,parallel(Ci, Cd));

Hs = feedback(series(PID, Gs),1);

stepplot(Hs, 0:0.01:10);

[st, t] = step(Hs, 0:0.01:20);