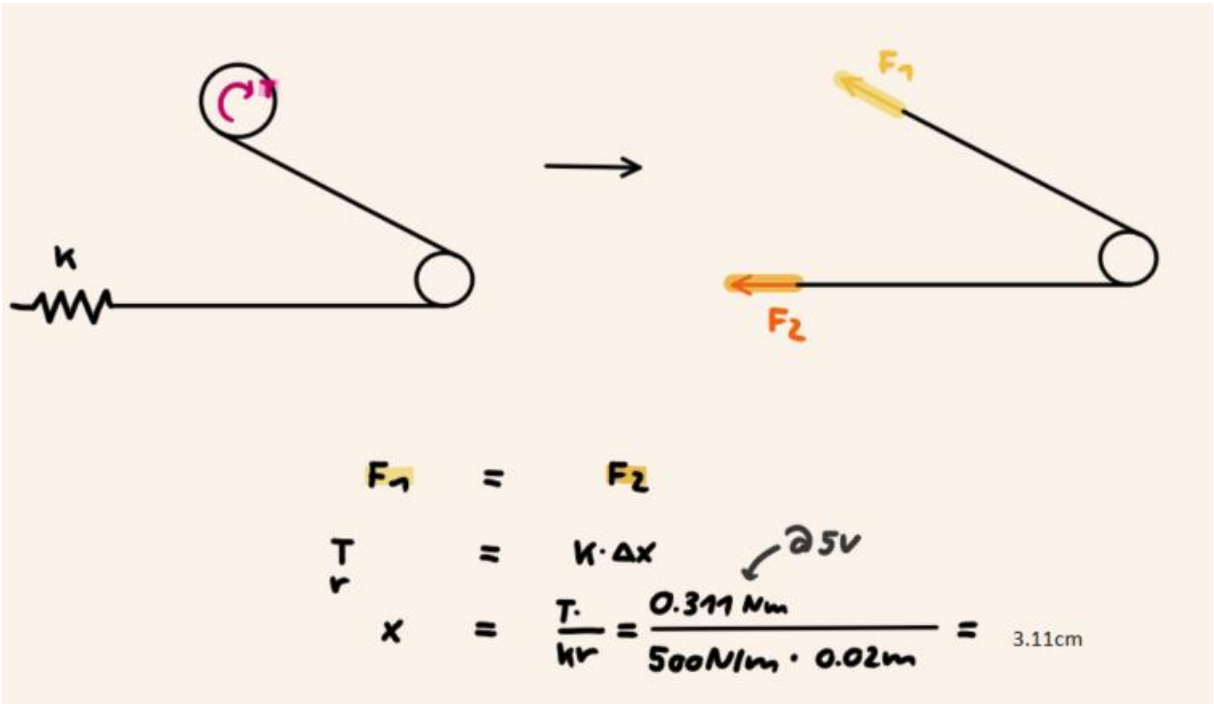
B4 Lab04

David Gögl, Ved Varshney, Emre Eryilmaz

PostLab Q3:

Measurments: 1.2cm which equals a efficiency of 1.2/3.1 ={approx} 40 %

Therefore, the displacement of the spring is directly proportional to the torque of the servo. If we know the maximum torque of the servo, we can calculate the maximum displacement of the spring before the servo stalls. [See calculation in PostLabQ2]



However, it's worth noting that this calculation neglects friction, which can cause losses and deviations in the system. Friction can arise from various sources such as the bearings in the pulleys, the air resistance of the blue string, and the internal resistance of the servo motor. These sources of friction can cause the actual displacement of the spring to be lower than the calculated value and can also cause deviations in the system behaviour over time.

PostLab Q4:

To increase the force that can be applied on the spring and achieve 1.5 times the deflection, we can use a gear train to multiply the torque output of the servo motor. The gear ratio needed will depend on the torque output of the servo motor and the desired torque needed to achieve the increased deflection.

If the original setup had a maximum deflection of x, and we want to achieve a deflection of 1.5x, we need to increase the torque by a factor of at least 1.5.

Using a gear train with a gear ratio of R, the new torque output T2 can be calculated as:

T2 = T1*R

Solving for R, we get: R = T2/T1 = 1.5

This means that we need a gear train with a gear ratio of 1.5 to achieve the desired deflection.

To implement the gear train, we can use a combination of gears with the appropriate tooth counts to achieve the desired gear ratio.

We set up the gears such that we got AT LEAST 1.5x the deflection, which was a combo of

$$(36/60) * (48/12) = 2.4$$

We got a deflection of 2.1, which is 1.75 times larger than the deflection we measured for Q3 (=1.2)

PostLab Q5/Q6:

We can only turn the servo by 180 degrees in 2.5mins (150s).

For the minutes side we need R = 2, as when the servo turns 180 degrees in 150s the other should make one full tunr.

1/2* 31.1Ncm = 15.55Ncm

The seconds timer should turn 150s/60s = 2.5 times so the R here should be 5

1/5 * 31.1Ncm = 6.22Ncm

PostLab Q8:

This is a simple function for implementing a delay in seconds using the clock() function provided by the C standard library.

The function takes an argument number_of_seconds that specifies the number of seconds to delay for. It first calculates the number of microseconds to delay for by multiplying the number of seconds by 1,000,000 (since there are 1,000,000 microseconds in a second).

Next, it gets the current time using clock() and stores it in start_time. The clock() function returns the number of clock ticks that have elapsed since the program started running.

The function then enters a loop that continues to execute until the current time, as returned by clock(), is greater than or equal to the start time plus the number of microseconds to delay for. Essentially, this loop continuously checks if the specified delay time has elapsed yet.

Once the loop exits, the function has completed the delay and the program will continue executing from where it left off.

It's worth noting that this function uses busy waiting, which means that the processor will be executing the loop continuously until the delay time is up. This can potentially waste processing power and should be used judiciously.

PostLabcQ10:

If we want to move by 180 degrees in 150 seconds, we have to move 1.2 degrees per second

If 550 us corresponds to o degrees and 2400us corresponds to 180 degrees, with a linear interpolation one gets: 562.33333333 periodically. So the difference is 12.3333333.