Lab#4 Servomotor – closed loop control

Leonardo Fusser (1946995)

Objectives:

- C code a PID algorithm to control the position of a BDC motor in closed loop mode.
- Fine tune a PID controller in simulation mode.
- Fine tune a PID controller in target mode.

Hardware: same as lab3.

To hand in:
On GitHub:

- Answer to all questions of this document on Teams.
- An indented and commented C code. See software requirement of the previous labs.

Requirements:

- You must write a closed loop PID controller to control the position of a motor (AKA servomotor control).
- The sampling time must be 10mS.
- To keep it real time, no blocking delays are allowed.

Set point:

- The slider pot will be used for the set point.
- The slider pot is connected to the ADC converter.
- The ADC converter has a 0-1023 range (10bits).

Program structure:
Your program should have the following structure:

```
int main(void){

	/* Initialize all required resources here */

	while (1){
		/* runs the PID controller - Close Loop mode */
		pid_control();
	}
}

void pid_control(void){

	if(TickDiff(stamp) > TICKS_TEN_MS){

		/* Do the pid calculations every 10mS here */

		/* Display info to LCD – Target mode only */
	}
}
```

Lab Work

In this lab, you are going to implement a closed loop control task.

GitHub:

- Clone the previous lab repository.

Part1: Simulation mode

As you notice, it is not easy to achieve a specific position PV. Therefore, a PID control loop is needed to attain this goal.

Implement the PID control loop by respecting a minimum of requirements:

- Loop sample time of 10mS.
- No blocking delays.

Now the ADC will be used to adjust the SP variable.
Therefore, change the scale of the ADC to obtain the following SP ranges:

Position 0% (left) SP = 0 tics.
Position 100% (right) SP = 320 tics.

Fine tuning in simulation mode

Once the PID control algorithm is up and running, set Kp to 30, Ki to 0 and Anti-windup to 0.

Using the potentiometer, quickly step increase the SP from 0 to 160.
Fill the column below.

	Kp = 30 Ki = 0 Anti-windup = 0
SP (tics)	160
SP (°)	~180°
Overshoot (tics)	53
Overshoot (°)	~60°
SSE (tics)	5
Settling time (mS)	~850mS

$$SP\ (°) = \frac{320\ tics}{160\ tics} = \frac{360°}{x} = \sim 180°\ per\ revolution$$

$$Overshoot\ (°) = \frac{320\ tics}{53\ tics} = \frac{360°}{x} = \sim 60°\ per\ revolution$$

Now that you know a little bit more about the behaviour of the system, fine tune it to minimize the error and the overshoot.

Keep Ki and Anti-windup to 0 and find the proper Kp by trial and error. Fill the table with some interesting results.

	Kp = 20 Ki = 0 Anti-windup = 0	Kp = 10 Ki = 0 Anti-windup = 0	Kp = 5 Ki = 0 Anti-windup = 0
SP (tics)	160	160	160
SP (º)	~180°	~180°	~180°
Overshoot (tics)	34	4	1
Overshoot (º)	~40°	~4.5°	~1.2°
SSE (tics)	3	5	22
Settling time (mS)	~770mS	~690mS	~690mS

Kp = 20:

$$SP\ (°) = \frac{320\ tics}{160\ tics} = \frac{360°}{x} = \sim 180°\ per\ revolution$$
$$Overshoot\ (°) = \frac{320\ tics}{34\ tics} = \frac{360°}{x} = \sim 40°\ per\ revolution$$

Kp = 10:

$$SP\ (°) = \frac{320\ tics}{160\ tics} = \frac{360°}{x} = \sim 180°\ per\ revolution$$
$$Overshoot\ (°) = \frac{320\ tics}{4\ tics} = \frac{360°}{x} = \sim 4.5°\ per\ revolution$$

Kp = 5:

$$SP\ (°) = \frac{320\ tics}{160\ tics} = \frac{360°}{x} = \sim 180°\ per\ revolution$$
$$Overshoot\ (°) = \frac{320\ tics}{1\ tic} = \frac{360°}{x} = \sim 1.2°\ per\ revolution$$

From your test, what is the best value of Kp? Explain why it is the best value.

- In simulation mode, it was determined that the best value of Kp was 10. This is because when Kp was set to 10, the lowest overshoot and settling time was achieved (lower is better in this situation!). Also, with a Kp set to 10, the second-lowest error (SSE) was achieved (Compared to the others).

Now that you know the best Kp value, don't change it.

By trial-and-error increase Ki and the Anti-windup until the SSE is one tic or less.

	Kp = 10		
	Ki = 10	Ki = 5	Ki = 0.1
	Anti-windup = 10	Anti-windup = 5	Anti-windup = 1000
SP (tics)	160	160	160
SP (º)	~180°	~180°	~180°
Overshoot (tics)	5	4	33
Overshoot (º)	~5.6°	~4.5°	~1.2°
SSE (tics)	5	5	1
Settling time (mS)	~670mS	~680mS	~2.740S

Ki = 10 & Anti-windup = 10:

$$SP\ (°) = \frac{320\ tics}{160\ tics} = \frac{360°}{x} = \sim 180°\ per\ revolution$$
$$Overshoot\ (°) = \frac{320\ tics}{5\ tics} = \frac{360°}{x} = \sim 5.6°\ per\ revolution$$

Ki = 5 & Anti-windup = 5:

$$SP\ (°) = \frac{320\ tics}{160\ tics} = \frac{360°}{x} = \sim 180°\ per\ revolution$$
$$Overshoot\ (°) = \frac{320\ tics}{4\ tics} = \frac{360°}{x} = \sim 4.5°\ per\ revolution$$

Ki = 0.1 & Anti-windup = 1'000:

$$SP\ (°) = \frac{320\ tics}{160\ tics} = \frac{360°}{x} = \sim 180°\ per\ revolution$$
$$Overshoot\ (°) = \frac{320\ tics}{1\ tic} = \frac{360°}{x} = \sim 1.2°\ per\ revolution$$

From your test, what is the best combined value of Kp, Ki and Anti-windup? Explain your choice.

- In simulation mode, it was determined that the best value of Kp, Ki and Anti-windup was 10, 0.1 and 1'000 respectively. This is because we were able to achieve the lowest error (SSE of around 1 tick) with those three values set. On the other hand, with those three values set, we also saw that it created the longest settling time and largest overshoot (compared to the others). Lowest is not better in this situation!

Part 2: Target mode

In target mode you must enable the following resources:

```
initIO();
initPWM();
initPV_measure();
initADC();
LCDInit();
```

Redo the same steps you did in simulation mode but on the target.

Now, you must display the setpoint SP along with the position PV on the LCD:

PV:
SP:

Like in target mode, the ADC will be used to adjust the SP variable:

Position 0% (left) SP = 0 tics.
Position 100% (right) SP = 320 tics (one full revolution).

Fine tuning in target mode

Enable the PID control task and then set Kp to 30, Ki to 0 and Anti-windup to 0.

Increase the SP value back and forth from 0 to 320 and from 320 to 0.

Give the worst-case error – the largest error that you measured.

Fill the column below.

	Kp = 30
	Ki = 0
	Anti-windup = 0
SP (tics)	320
SP (º)	~360°
SSE (tics)	20

$$SP\ (°) = \frac{320\ tics}{320\ tics} = \frac{360°}{x} = \sim 360°\ per\ revolution$$

Now that you know a little bit more about the behavior of the system, fine tune it to minimize the error and the overshoot.

Keep Ki and Anti-windup to 0 and find the proper Kp by trial and error. Fill the table with some interesting results.

	Kp = 50 Ki = 0 Anti-windup = 0	Kp = 70 Ki = 0 Anti-windup = 0	Kp = 300 Ki = 0 Anti-windup = 0
SP (tics)	320	320	320
SP (º)	~360°	~360°	~360°
SSE (tics)	8	5	1

For all columns above:

$$SP\ (°) = \frac{320\ tics}{320\ tics} = \frac{360°}{x} = \sim 360°\ per\ revolution$$

From your test, what is the best value of Kp?

- While in target mode, it was determined that the best value of Kp was 300. This is because we achieved the lowest error (SSE of around 1 tic) when Kp was set to 300 (compared to the others). In this case, lowest is better!

Now that you know the best Kp value, do not change it.

By trial-and-error increase Ki and the Anti-windup until the SSE is one tic or less.

	Kp = 300		
	Ki = 300 Anti-windup = 300	Ki = 50 Anti-windup = 100	Ki = 0.1 Anti-windup = 200
SP (tics)	320	320	320
SP (º)	~360°	~360°	~360°
SSE (tics)	5	2	16

For all columns above:

$$SP\ (°) = \frac{320\ tics}{320\ tics} = \frac{360°}{x} = \sim 360°\ per\ revolution$$

From your test, what is the best combined value of Kp, Ki and Anti-windup? Explain your choice.

- While in target mode, it was determined that the best value of Kp, Ki and Anti-windup was 300, 0.1 and 200 respectively. This is because it was shown that the system produced the least number of oscillations, despite have the worst error (SSE of around 16). In this situation, lowest also doesn't necessarily mean better!

When the motor is stabilized at a specific position, take a screenshot of **Dir** and **En**.

- Refer to “Figure 1” below.

Approval before dismantling!

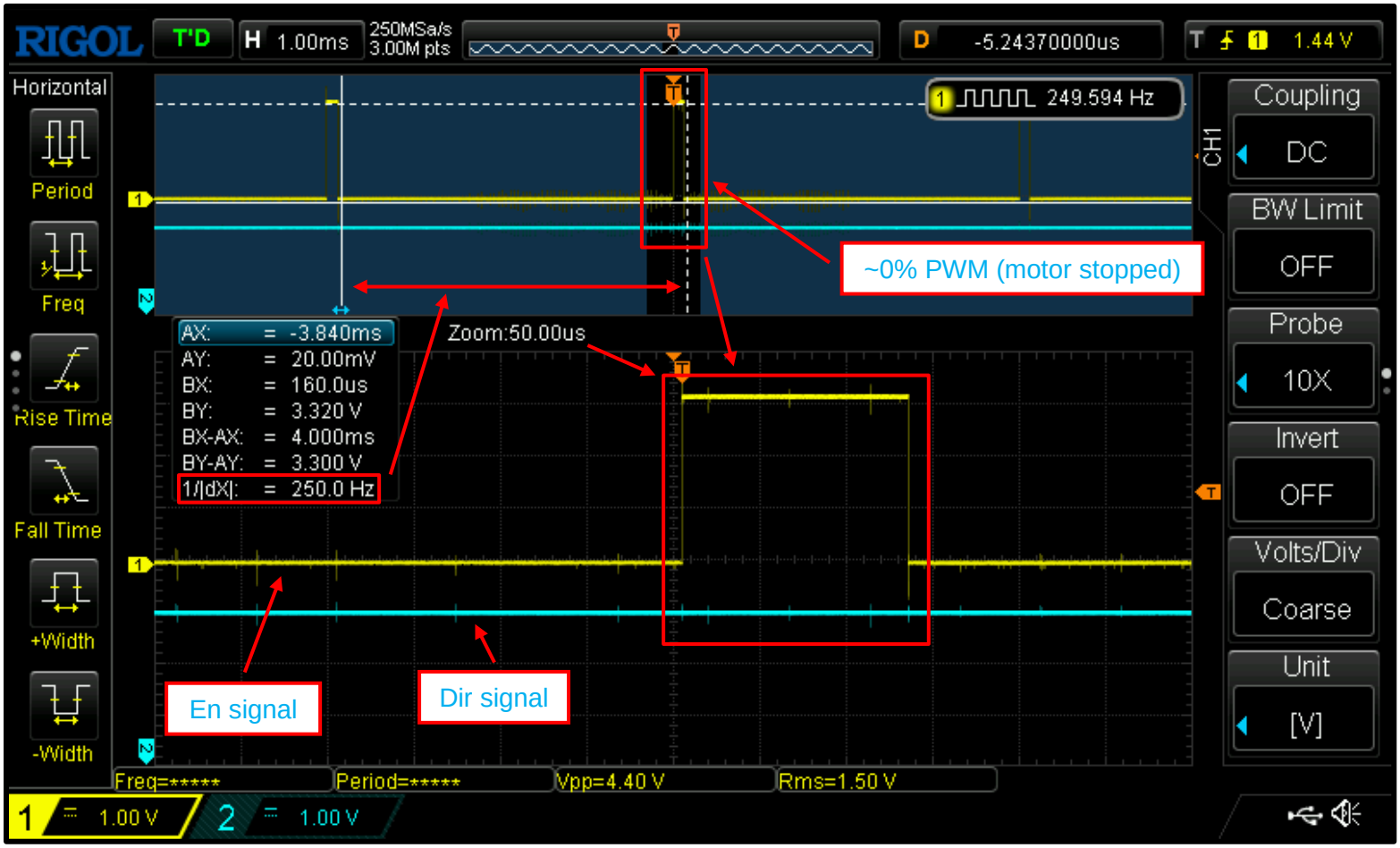

Figure 1. Screenshot for Dir (pin G9 on PIC32) and En (pin G8 on PIC32) signals shown above. Since this screenshot was taken when after the motor stabilized, the En signal should show a PWM close to 0%. As we can see above in the En signal (yellow), this is indeed the case. The Dir (blue) signal shows high because the motor stabilized after the SP was set from 0 to 320 (this is the forward direction – reverse shows opposite effect of forward).

After lab questions

1- Explain what you learned in this lab. What went wrong and what did you learn from your mistakes. Give specifics please.

- Similar to the previous lab, the main issue at hand here was that I did not quite understand the fine details behind the operation of a closed-loop control system (coding was not an issue). For example, I did not know how values such as P-term, I-term, Anti-windup affected how the system would produce an output (done through PV value). I also did not know what was considered as a good result and a bad one (for example, I did not know that large oscillations in the system's output were not always necessarily a good thing). Once some deeper research and experimentation was done, I was able to overcome these issues I encountered.

2- In simulation mode, what was your best results in term of Kp, Ki and anti-windup?

- While in simulation mode, the best results in terms of Kp, Ki and Anti-windup were 10, 0.1 and 1'000 respectively. This was because we were able to achieve the lowest error (SSE of around 1 tick) when these three values were used.

3- In target mode, what was your best results in term of Kp, Ki and anti-windup?

- While in target mode, the best results in terms of Kp, Ki and Anti-windup were 300, 0.1 and 200 respectively. This was because we were able to see the lowest number of oscillations being created by the system when the slider pot was changed on the Explorer 16/32 board. This especially matters in applications such as robotic arms in assembly lines where the movements of the arms cannot have any unnecessary movements (oscillations), otherwise the operation of the arms are un-reliable!

4- From the previous two questions, do you think the BDC motor model has the same characteristics as the real motor?

- Based on the answer given in the previous two questions, we can say for a fact that the BDC motor model does not have the same exact characteristics as the real motor. There are many reasons as to why this may occur, one of the obvious ones being that not all motors are the same. For example, if a simulation model was created based off a real DC motor, the results may differ if another type of motor (ex: stepper motor) was used to test functionality. We are not certain if the motor model provided to us was based off of the real DC motor that was used in this lab. The other obvious answer would be that it is very difficult to create a simulation model that matches the characteristics of any given BDC motor, and this could also be for many reasons.