TASK 5 Optical Switch & feedback to Stepper motor

A. Problem Statement:

Integrate and calibrate optical switch with the test jig for real-time velocity, RPM, and position feedback; develop closed-loop control using PID and mathematical models for increased accuracy and precision.

B. Sub-tasks Completed:

1. Set up and calibrated optical switch, encoder.
2. Built and tested sample pulse count code for optical encoder testing.
3. Interfaced Stepper motor and optical switch encoder by building and testing code for counting pulses and distance travelled
4. Added & merged distance mode as stated above with existing final merged code of Stepper motor.
5. Studied about feedback and real time closed loop systems.
6. Built and integrated the merged final code with PID (Kp,Ki,Kd) feedback and tuning.
7. PID code tested successfully – apart from PID tuning of constants.
8. Developed and studied for second feedback using mathematical equations between Steps of Stepper motor and pulses counted by optical encoder.
9. Took data readings for developing and formulating the feedback equation.
10. Formulated and studied feedback equation.
11. Built and tested code using real time feedback equations (Quadratic, linear models).

C. Efforts Details

Sr. No.	Activity Name	Details	Date	Remarks
1	Optical_switch setup	Setup & testing of Optical switch. Sample test code built and tested to monitor real-time velocity & RPM	3 July	Optical switch testing and setup. Calculations and calibration done. Sample code tested to read real-time velocity and RPM of the object
2	Feedback study & implementation	Study of guide of PID closed loop code struct and calibration. Built code for the same	3 July,4 July	Study on PID controller. Built PID control code and merged with main code
3	Testing & calibration of accelerometer	Testing and calibration of accelerometer using test jig. Built code to count number of pulses & distance travelled for user input steps. Observations and formulation of equation	5 july	Code tested successfully. Took readings from 10 steps to 32000 steps. Data analyzed to derive mathematical relation between steps and pulse
4	Final feedback	added mode 5 (distance mode) and merged with final code. 2) Added quadratic feedback equation for closed loop and real-time monitoring of velocity, acceleration, pulse, error	5 July	Distance mode tested for user input distance in cm. Feedback equation built and tested. Few errors need correction for improved accuracy

D. Observations / Learnings:

1. Optical encoder is based on pulse count mechanism and uses the same to calculate the velocity, rpm, distance and other parameters.
2. The stepper motor and wheel (placed between the optical sensor is in **1:4 ratio**.
3. The encoder counts **16 pulses in 1 wheel revolution** & thus **64 pulses for 1 stepper motor revolution.**
4. **5 stepper motor revolution => 80cm distance travelled by object.**

5. PID STUDY:

⇨ **PID Feedback Control (Proportional–Integral–Derivative):**

A **PID controller** continuously calculates an error value as the difference between a desired setpoint and a measured process variable, and applies a correction using three terms:

a. **Proportional (P) – Kp × Error**

- Reacts to **present** error.
- Larger Kp → stronger, faster response.
- Too high Kp → overshoot or oscillation.

b. Integral (I) – Ki × ∫Error

- Reacts to **accumulated past** errors.
- Eliminates steady-state error (offset).
- Too high Ki → slow response, instability.

c. Derivative (D) – Kd × d(Error)/dt

- Reacts to **rate of change** of error.
- Predicts future trend → smooths response.
- Too high Kd → noise amplification.

⇨ **Final Control Output:**

$$\text{Output} = Kp \cdot e(t) + Ki \cdot \int e(t)dt + Kd \cdot \frac{de(t)}{dt}$$

Where e(t) is the error at time t.

⇨ **Summary:**

Term	Purpose	Acts on
Kp	Proportional gain	Present error
Ki	Integral gain	Past errors
Kd	Derivative gain	Future trend

6. Built and tested code with PID feedback – code worked but tuning needs to be done for kp,ki,kd to reduce oscillations
7. Designed and developed systematic tests to be done to tune PID
8. PID tuning feedback changed to equation feedback as per project requirements and specification
9. Built and tested code to count pulse for user defined steps.
10. Took readings for steps and pulses and plotted graph of steps,pulses and distance
    - ⇨ Steps Vs Distance is a linear graph indicating both vary linearly
    - ⇨ Steps Vs Pulses is a rising graph can be characterized by both linear and quadratic equations
11. Designed and developed mathematical model based on steps pulse data . (for reference refer : https://drive.google.com/drive/folders/1hxZ6KVwZZAAeKOxUEeYY0fXFS5TXSw09)

a. Quadratic model analysis :

   ⇨ The quadratic term in equation is used to eliminate the error which occurs at high step rate
   ⇨ Implementation using this lead to high positon error and large difference between ideal and actual readings
   ⇨ After many testings,it lead to conclusion that quadratic model is not compatible with our test_jig setup

b. Linear model:
   ⇨ To simplify the model and align with our practical test jig setup shifted to linear model (Pulse = k* steps ; k = constant)
   ⇨ Code built and tested worked but value of k needs to be adjusted according to real readings and observations.

12. To get accurate readings – built code to:
    a. develop closed-loop feedback mathematical model by continuously correlating the stepper motor's commanded step position with real-time optical pulse feedback.
    b. By capturing data every 100 ms during motion—including steps taken, expected vs. actual pulses, and positional error—it quantifies the relationship between mechanical motion (steps) and sensor feedback (pulses).
    c. In short, using real time motion profile data of object of our test jig to develop feedback equation this which allows precise modelling of system accuracy, detection of missed steps or lag, and validation of motion profiles.

d. Data printed:

Parameter	Meaning	Unit	Formula used
Mode	Current motion mode	-	
T	Time since startup	Milli sec	

Step	Absolute position from soft zero (user - defined 0 position)	Steps	long stepsTaken = currentStep - motionStartStep;
(rel)	Steps moved since the motion started	Steps	
Dist	Distance moved (steps*cm per step)	Cm	float stepDistance = abs(stepsTaken) * CM_PER_STEP;
Pulse	Number of valid optical pulses detected	Pulses	pulseDistance = currentPulseCount * CM_PER_PULSE;
(exp)	Expected pulses (based on steps*pulses/step)	Pulses	float expectedPulses = abs(stepsTaken) * PULSES_PER_STEP;
Deficit	Difference between expected and actual pulses	Pulses	float pulseDeficit = expectedPulses - currentPulseCount;
Pdist	Distance via pulse count (pulses*cm per step)	Cm	
Err	Difference between step-based and pulse-based distances	Cm	positionError = stepDistance - pulseDistance;
Vel	Current velocity (stepper speed converted)	m/sec	float velocity_ms =abs(stepper.speed()) / STEPS_PER_METER; float acceleration_ms = stepper.acceleration() / STEPS_PER_METER;
MinInt	Minimum time between any two valid pulses	Micro -sec	
(Hz)	Maximum pulse frequency observed during motion	Hertz (Hz)	float maxPulseRate = 1000000.0 / currentMinInterval; // in Hz

13. Final Feedback code ready , little minor issues to be resolved.

E. Deliverables

⇨ https://drive.google.com/drive/folders/1vN_qomplQiLGPzCcW-l0SwOjxoYLMVAY

F. Conclusion

- Distance mode added and merged code with all 5 modes without feedback ready
- Merged final code with PID feedback built and tested (only tuning if needed can be done)
- Merged final code with real time data of parameters with errors and deficit between actual and calculated built for debugging and detailed analysis
- Final feedback code built and tested successfully.