Sebastian Zapata
CS 513
Winter 2024

Robust Control Semester Project

Ball-beam System Control

Introduction

A big focus of this course has been to understand how to deal with instabilities and perturbations in a given system, by applying control techniques that result in being able to stabilize the behavior of it towards a specific target. Some of those techniques were LQR and LQG controllers, which implement robust control. However, these techniques require more analysis, mathematical tools, and information about the system dynamics and states. A specific control strategy not covered in this course is Proportional-Integral-Derivative control (PID) which though not as sophisticated, provides more flexibility and requires minimal knowledge about the system dynamics to provide decent performance.

The objective of this project is to explore in more detail both the theory and implementation of the PID control algorithm and – if time permits before the semester is over – compare it to the performance of the LQR controller. This will be done by both showing the theory behind PID and then implementing them in hardware with a custom-made ball-beam system. A section will be dedicated to describing the theory of the PID algorithm with its variations depending on the knowledge available of the system being controlled, followed by a description of the demo built to test the algorithm. Afterwards, the dynamics of the system will be derived, and a qualitative analysis of the performance will be shown.

A significant portion of the work for this project went into the building, assembly, coding, and testing of the demo, which cannot be included in the report, however a reference section with links to the code and pertinent files for fabrication and assembly can be found at the end.

Background: PID

The PID control algorithm relies on being able to measure the state of a target parameter in the system and receiving a target/desired value for such parameter. This allows the algorithm to calculate the mismatch between them (error) and adjust the input to the system so as to modify the state of the given parameter and bring it closer to the target value. Figure 1 shows a block diagram of this strategy, and as it can be noted, the error is passed through a proportional block, an integral block, and a derivative block, each of which contributes to the input signal that will be passed into the “process” block that encapsulates the behavior of the system.

Sebastian Zapata
CS 513
Winter 2024

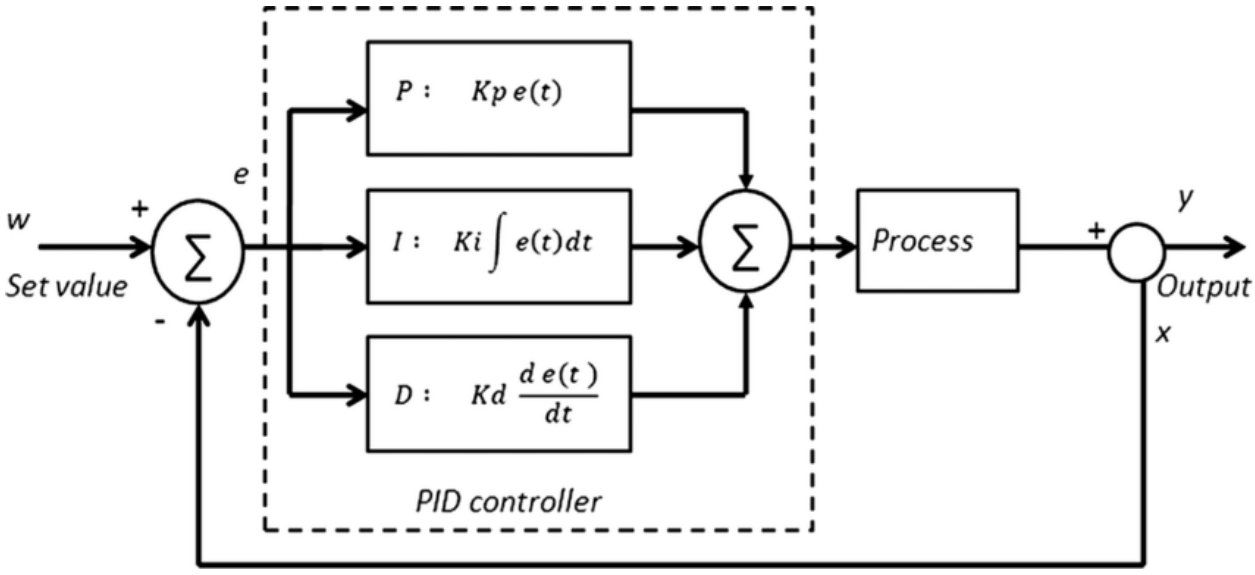

Figure 1. PID block diagram, highlighting the proportional, integral, and derivative blocks that take the error signal and provide an input signal to the system

Each of the controller blocks has a specific functionality in this algorithm, and they are specified below:

1. *Proportional (P):* a constant K_p multiplies the error signal, which means that the bigger the error signal, the bigger the proportional response to it
2. *Integral (I):* depending on the behavior of the system, error in the state can be perpetuated over time in what is called "steady-state error". To solve this problem, the value of the error can be tracked over time and integrated and multiplied by K_i, so the larger the "accumulated" error, the larger the compensation for it
3. *Derivative (D):* depending on whether the error signal changes slowly or rapidly, ideally, we would want out controller to react correspondingly as well, so in this block we calculate the derivate of the error signal by tracking past values and observing how fast they are changing. The derivative of the error signal is then multiplied by K_d . This allows us to account for the speed at which the error signal changes.

In the sense described above, each of the constants or "gains" (K_p, K_i, and K_d) become independent knobs we can use to modify the response of our controller to the errors in the system. Since every system behaves differently, finding the right combination of "gains" that will ensure good performance given a desired value for the states is a process

that can take significant amount of work. This process is called “tunning” and even though there are many guidelines and algorithms that have been developed to efficiently perform it (which are not discussed in this work), it is by nature a trial-and-error process until a “good enough” combination is found. In this sense, there’s no perfect tunning, only better or worse combinations of gains; this decision is therefore under the prerogative of the system operator.

It is worth noting that this diagram very abstractly denotes the “process” block as a simple black box that takes the input signal and provides an output signal but doesn’t specify any kind of structure or requirement. This is one of the big advantages of PID with respect to other control techniques: prior knowledge of the system dynamics is optional. Knowledge of the dynamics of the system enriches the control performance as the behavior becomes even more predictable, however, this is not strictly needed as a good tunning process will still offer a good response to a desired behavior in the most general case. It is, however, plausible that certain systems will still perform poorly despite a decent amount of tunning work being performed, so by adding dynamics to the algorithm or even considering other control techniques, more ***robustness*** can be introduced.

Finally, the diagram in figure 1 shows each of the operation blocks as if the error signal was in continuous time, however, since this algorithm will be implemented in software, the integral and derivative blocks will instead be changed to their corresponding discrete time counterparts.

Demo Description

The tool built to implement the algorithm is a combination of a 3D printed ball-beam system with an actuator, sensors, a microcontroller and supporting power system integrated with a web app for manual control, feedback control manipulation, and live visualization of the sensor readings. Figure 2 shows the project organization and the integration of hardware and software required to perform tests of the PID algorithm. It also shows the communication protocol (or controlling signal scheme) used for the interaction of all the connected blocks.

The hardware realization of the system is shown in Figure 3, along with the main software tools employed in the development process of the GUI and communication of the individual components of the system.

Sebastian Zapata
CS 513
Winter 2024

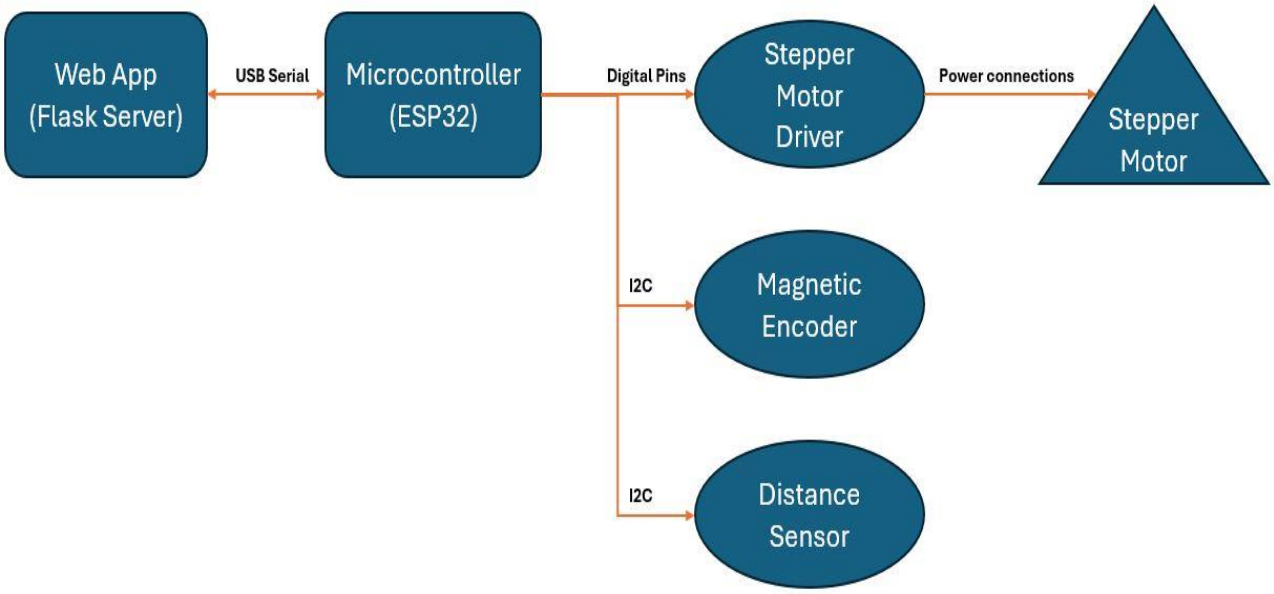

Figure 2. Block diagram describing the integration of hardware and software of the project

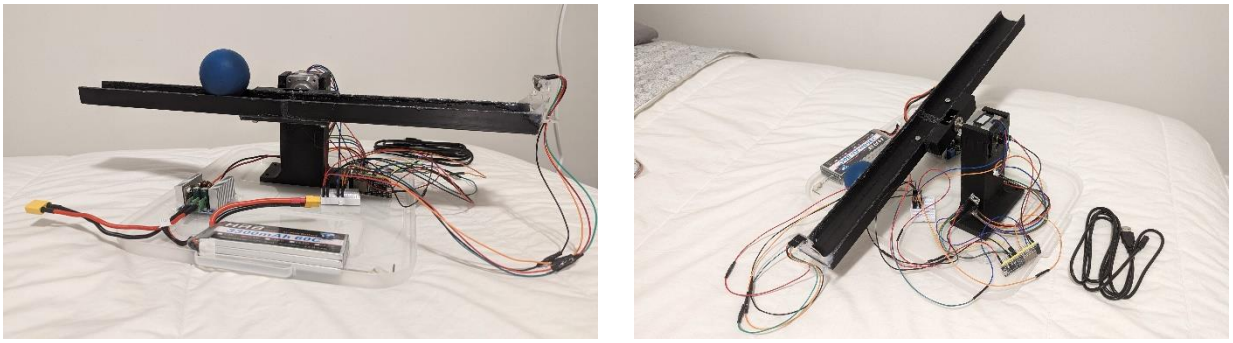

Figure 3. Hardware realization of the integration and software tools employed for communication and control of the hardware

It should be mentioned that a non-negligible amount of work went into the calibration of the sensors in the system. For the magnetic encoder, a good understanding of the

mechanics of the process for angle recognition was needed to effectively include the sensor as depending on the orientation of the magnet on the motor shaft, different initial angles are read. It also should be noted that due to unexpected behavior at initialization, different angles are read every time, so a “zero out” functionality was built in so that the operator can define the zero position. After this, the sensor performs as expected indicating angles relative to the user-defined zero position.

For the distance sensor, different positions and orientations had to be tested (not to mention different kinds of sensors had to be tested) to make sure an optimal and least error-prone measurement could be guaranteed within the range of interest for the positioning of the ball.

Performance Assessment

After assembly, manual testing, and calibrations were performed, the algorithm was loaded on the microcontroller to perform control of the ball position on the beam provided certain target values coming from the web app over serial communication. A qualitative assessment was initially conducted to make sure the ball position followed the desired value and then a more quantitative assessment was conducted to measure the response accuracy and how long it took for the system to reach the desired value. At the time of writing of this initial draft, tests haven’t been completed, however they will be performed as described above. An expected effect is the one of noise in the distance and angle sensors which will likely introduce minor to major instabilities that may prevent the system from reaching adequate behavior. This and other observations will be included in the final draft.

Conclusions

PID is an excellent control technique for simple and even complex systems and provides significant flexibility with respect to other control techniques such as LQR or LQG. However, it suffers from lack of accurate and/or prompt response depending on the quality of the tunning and lack of guaranteed performance as the noise/disturbances on the sensors is not accounted for in the mathematical structure of the controller.

For applications sophisticated guarantees on performance aren’t required, PID is an excellent way to offer control capabilities with low modelling and computing requirements.

Sebastian Zapata
CS 513
Winter 2024

Source Materials

- GitHub repository – *[link not yet published]*
    - Flask web app project
    - Microcontroller firmware
    - STL files for ball-beam system
    - Pertinent datasheets