Vineet Tambe

Team H: DockDockGo

Teammates: Sushant Jayanth, Siddhant Wadhwa, Sergi Widajaja, Soham Bhave

ILR01

Feb 8 2023

1. Individual Progress

For the Sensors and Motors Lab – I was tasked with integrating a potentiometer, integration of a push button to manage states of the system, developing the PID control for velocity control for the 12V DC motor, integrate all the individual pieces into one unified functioning code and wire the circuit for the same such that it works reliably, integrate the Arduino code with the GUI.

1. The potentiometer is a basic 3pin voltage divider which gives a range of voltage in between the two voltage levels across its terminals. The reference voltage used for this particular lab was 5V at one terminal and 0V (Ground) at the other. The voltage at the third pin is measure using the analog to digital converter (ADC) of the Arduino. The Arduino’s ADC has a range of 0 to 1023 which needs to be mapped to 0 to 5V in order to get the accurate voltage across the wiper pin of the potentiometer. The circuit architecture shown in figure 1.1 was used for the potentiometer.

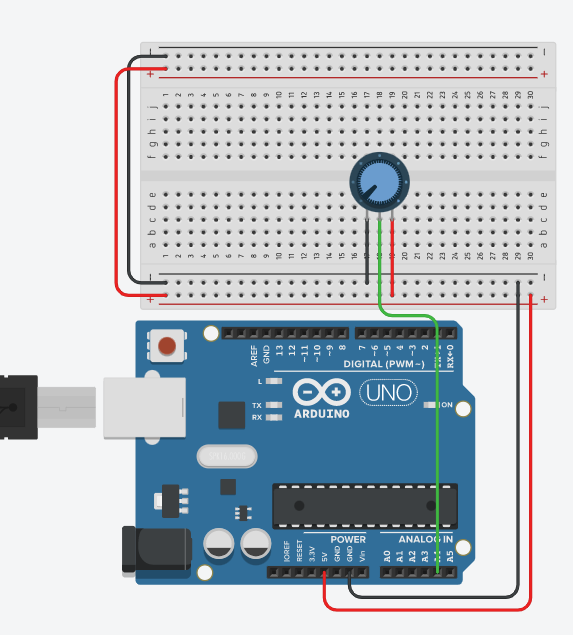


Figure 1.1

To obtain the transfer function of potentiometer we plotted a graph of the angle of the wiper on the x-axis and the voltage value across the wiper on the y-axis. We did so by rotating the wiper in increments of 10 degrees and noting down the voltage value using a digital multi-meter. This was done in both directions, i.e., 0 to 300 (maximum allowable angle of the potentiometer) and back from 300 to 0 degrees. There was a non-linearity and hysteresis in the values obtained as seen in the Figure 1.2

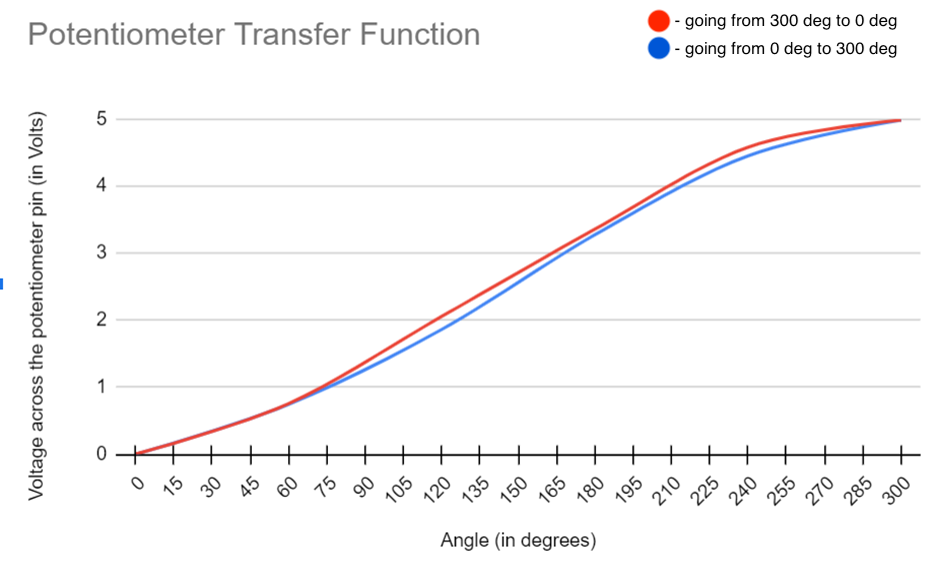


Figure 1.2

1. The push-button is a simple switch which is used to trigger Arduino UNO’s hardware interrupt pin 3. The circuit diagram shown in figure 1.3 is used to connect the push-button to the Arduino. The resistor used is a 220 Ohm resister.

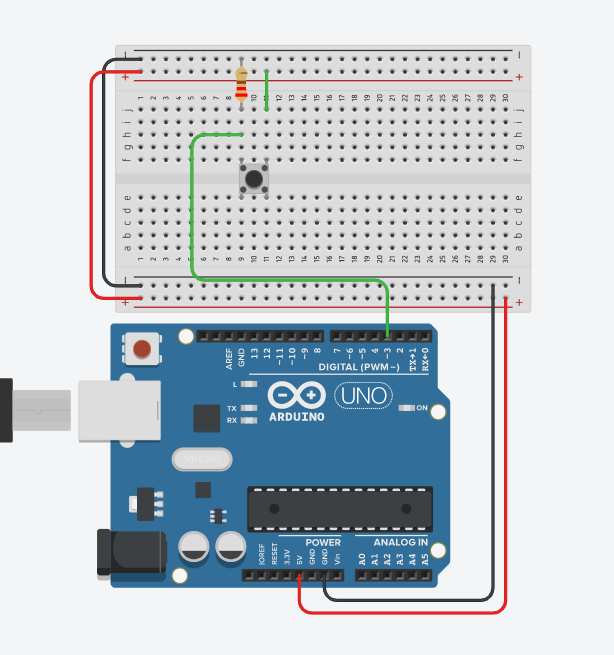


Figure 1.3

1. A debounce timeout of 200 milli seconds is used to prevent any false triggering or retriggering of the push button. This push button also turns the built in led at digital pin 13 of the Arduino in order to give visual feedback to the user of the state of the system.
2. For the DC motor velocity control a simple PID controller was implemented and tuned on the real setup as opposed to the previous development done first on simulation.
3. After everybody was done with their assigned tasks the code was finally integrated and tested on the entire circuit. This was a smooth process as standard modular coding standards and formatting was followed while writing our individual components.
4. To integrate with the GUI a strategy to communicate the sensor and motor information had to be developed. The architecture we ended up devising is as follows:
   1. The GUI send a Unicode 1 byte character ‘r’ to read the sensor data and system states. And the character ‘w’ to give commands to the actuators.
   2. When ‘r’ is received a 4 byte array with every byte carrying information about the sensor is written in the following order-
5. Challenges

During the course of this lab, I faced the following challenges:

1. Push Button – The push button, a simple component of the circuit, worked perfectly when tested individually however when integrated with all the other components the push button was very sensitive to noise. During this time the push button was connected in such a fashion that pushing the button directly connected the pin3 to 5V. This was modified such that the pin3 was connected to a 220 ohm resistor and ground at one end of the push button and the other end of the push button was connected to 5V. This improved the reliability of the push button and made it less susceptible to noise. Additionally, the debounce time had to be finely tuned for the circuit.
2. PID velocity control – Since the DC motor provided used highly geared planetary gearbox, the initial input voltage required for the DC motor to be set in motion was quite high (a pwm signal of 100 was necessary). Once the motor shaft was set in motion, the required pwm to maintain that motion goes down. Accommodating this non-linearity with PID controller was extremely tricky. A lower bound on the possible rpm which the controller can track had to be set which was experimentally found to be around 20 rpm. Any rpm above that could be stably tracked and anything below caused jerky motions because the shaft stopping and going.
3. Teamwork

The team was able to delegate tasks in such a way that nobody was overloaded with any particular work. We had initial meetings where we discussed various ideas to implement logic, code standards and architecture with the aim of keeping integration as easy as possible. This allowed us to have a clear vision of how the final result looked like.

Siddhant, Sergi, Soham, Sushant: As I was tasked with the integration of the code, I had to communicate with all the team members. In order to do this efficiently I used git to manage the project workflow and merge the updates to the codes of individual modules raised by every team member as we progressed through the lab.

Sushant: I was particularly impressed by the serial communication strategy in order to communicate with GUI, developed I coordination with Sushant. The idea was to send concise packets of byte array with information requested by the GUI.

1. Plans