- Lab report -
Lab 02: Force Sensor

[S26] Sensors and Sensing

- Innopolis University -

Anas Khmais Hamrouni
Vladimir Toporkov
Andrey Krasnov
Askar Kadyrgulov

B23-RO-01

[Lab notebook]

[Lab Solution Git Repository]:

**experiments steps/instructions in README*

Topic :

Setup, calibration, and application of a force sensor for impedance control of a BLDC motor using an ESP32 microcontroller.

Purpose:

The purpose of this work is to study a force (stretch) sensor and its amplifier, perform calibration and modeling of its measurement, and then apply the calibrated sensor in an impedance control task where a BLDC motor behaves as a virtual spring whose torque (or position) is proportional to the applied force.

Description of the sensor application

The force sensor converts applied load into an electrical signal that is amplified and read by the ESP32 as digital values via the JY-S60 amplifier. These readings are calibrated into force in Newtons using a linear model based on known weights and then used as the input for an impedance (virtual spring) controller that changes the BLDC motor angle proportionally to the applied force.
In the final setup, the sensor is mounted on the motor lever so that the dependence of force and motor current on the rotation angle can be recorded and analyzed to characterize the effective stiffness of the virtual spring.

List of used equipment/stack:

- **Hardware:** (documentation & datasheets in lab notebook)
  - Force (stretch) sensor ZNLBS-VII-10kg
  - Load-cell weight transmitter/amplifier JY-S60, used to amplify and condition the sensor signal for the ESP32 analogue input.
  - BLDC motor MF4005 V2 (datasheet)
  - ESP32-WROOM-32U microcontroller board, used for ADC acquisition and control implementation.
  - CANable CAN <-> micro usb adapter
  - Power supply
  - usb cables, wires, breadboard
  - 3d printer
  - Mechanical lever of length 0.125 m
- **Software tools:**
  - IDE
  - esptool: firmware flashing (Micropython)
  - rshell: code upload
  - picocom: data logging

Description of experiments

1. Force sensor reading, zeroing, and preprocessing

The ESP32 continuously reads lines from the serial port using `read_force_line()`, extracting the latest numeric value while rejecting empty lines, parsing errors, and unrealistic jumps. The raw value `F_raw` is held for up to `FORCE_HOLD_S = 1` s if new data do not arrive and is otherwise reset to zero; then a first-order low-pass filter `Ff = lowpass(Ff, float(F_raw), FORCE_LP_ALPHA)` with `FORCE_LP_ALPHA = 0.75` is applied to suppress noise. A deadband is introduced (`F_DEAD = 300.0`) so that effective force `F_eff` is set to zero below this threshold, which corresponds to practical zeroing of the sensor around no-load conditions.

2. Calibration and conversion ADC → Newton

The calibration is embedded as lookup arrays `ADC_POINTS` and `F_POINTS_N`, derived from known weights (0, 0.5, 1.0, 2.5, 3.0, 3.5 kg mapped to 0, 4.903325, 9.80665, 24.516625, 29.41995, 34.323275 N). The function `adc_to_newton(adc)` performs linear interpolation between these points, while clamping outside the range, which effectively models sensor nonlinearity and provides a measurement model of the form $FN=f(\mathrm{ADC})$. During experiments, the logged pairs (ADC, FN) are used to evaluate accuracy (difference between calibrated force and nominal weight) and hysteresis (differences between loading and unloading sequences, if such runs are recorded separately).

3. Mode logic and virtual spring behavior (demo_link)

The filtered force `Ff` is used to switch between two modes: `FORCE` and `RETURN`, with hysteresis thresholds `F_ON = 1000.0` and `F_OFF = 500.0`. When `F_eff` rises above `F_ON`, the controller enters `FORCE` mode, mapping the force to a normalized value $f01=\mathrm{clamp}(F\mathrm{eff}/F\mathrm{MAP\_MAX},0,1)$. A square root nonlinearity `g = f01 ** 0.5` and a maximum force-driven velocity `V_FORCE_MAX = 40.0` deg/s define the velocity command `v_cmd`. This velocity is rate-limited (`A_FORCE_MAX`) and integrated into the desired angle `q_des`, which is clamped to a maximum excursion `XMAX_DEG = 1080.0`. The virtual spring has a nonlinear stiffness profile between applied force and motor angle.

In `RETURN` mode (when `F_eff` falls below `F_OFF`), the motor is driven back to the home angle `q0` using a virtual spring with gain `VIRT_K_HOME = 2.0` and saturation (`V_MAX_RETURN`, `A_MAX_RETURN`), so the lever smoothly returns to its initial position once the user releases the force.

4. Motor current control and data logging (demo_link)

On each control cycle at `CTRL_HZ = 250.0` Hz, the program computes the position and velocity errors for a PD controller: `I_cmd = Kp * (q_des - q) + Kd * (dq_des - dq)` with `Kp = 3.0`, `Kd = 2.0`, and sets the motor current through

`motor.set_current(I_cmd)` after clamping to the allowed range. The script logs to `run_log.csv` at `LOG_HZ = 250.0` Hz the current time, motor angle `q_deg`, speed `dq_deg_s`, angular velocity `omega_rad_s`, commanded current `I_cmd_A`, raw ADC force value, calibrated force `F_N`, torque $\tau_{\mathrm{Nm}}=F_N \cdot LEVER_M$, and the current mode (`FORCE` or `RETURN`).

Description of the results of the experiments:

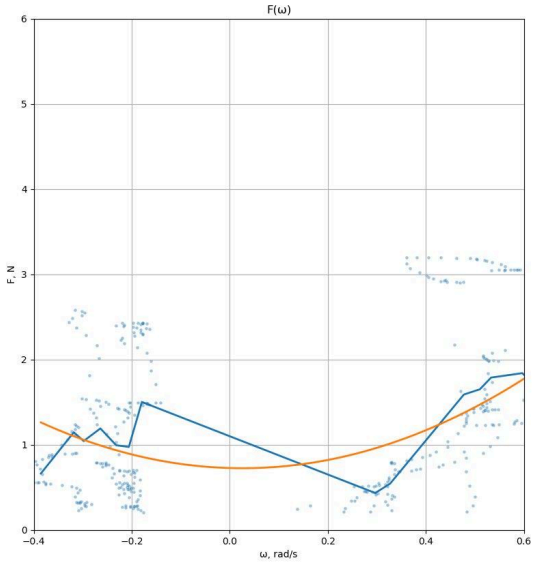
Using the calibration table and interpolation, we get a mapping from raw ADC values to physical force in Newtons, with the ADC range 0…~26000 corresponding to approximately 0…34 N based on the calibrated mass points.

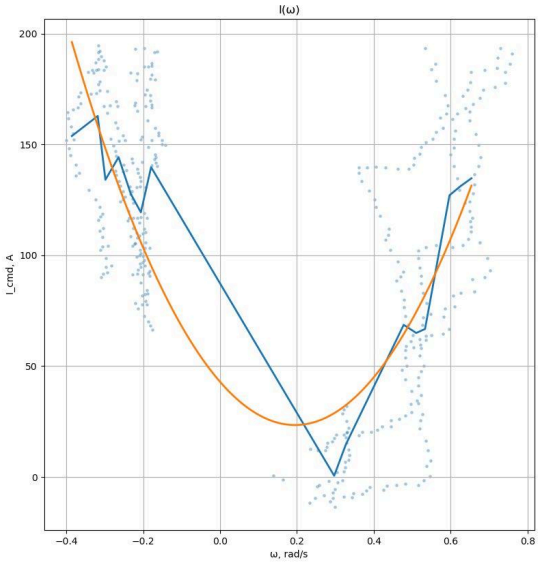
The deadband and low-pass filter noticeably improved signal stability during small or slowly varying loads.

The hysteresis in mode switching (`F_ON`/`F_OFF`) successfully prevented rapid toggling between `FORCE` and `RETURN` modes around the threshold showing smooth transitions in logged mode states and in the observed lever motion.

In `FORCE` mode, the motor angle `q_deg` in the log increased monotonically with growing force until reaching the software limit `XMAX_DEG`, while in `RETURN` mode the angle decayed back to `q0`, which visually corresponds to the video where higher pressing force produces larger motor rotation and release causes automatic return.

The recorded values of `I_cmd_A` and `tau_Nm = F_N * LEVER_M` versus `q_deg` showed approximately spring-like behavior (increasing torque and current with angle), with deviations attributable to sensor nonlinearity, calibration error, and motor friction, in line with the qualitative expectations of a virtual spring implemented via current control and the given calibration model.





* Light blue dots: Raw filtered data at each angular velocity

* Blue curve: Binned averaged values, mean value between bins

* Orange curve: 2nd-degree polynomial fit to the binned data