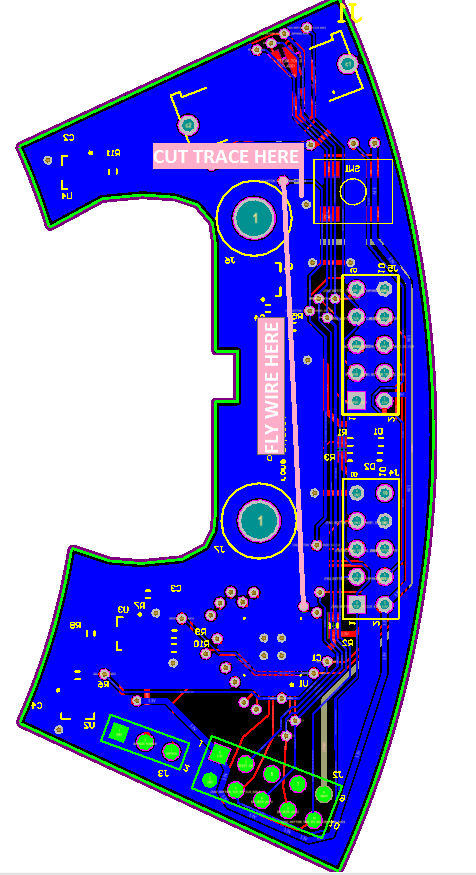
Gripper Board Overview

The gripper board is designed to be used with the mechanical gripper. It controls a single servo which is used to open and close the gripper so that objects can be grabbed. It features a current sensor on the servo which provides a direct correlation to force. Software is used force limit the gripper to prevent damage to the servo, object, or gripper. Because we do not want to break our cables or robots, a hard stop on the griper limits the rotation of the gripper to just under 360 degrees. The board also has two hall effect sensors mounted at either end. There are magnets on the mechanical gripper that will trigger the sensors so that the board can inform the robot it has reached the hard stop and so the robot can rectify the behavior by turning the opposite direction from the hard stop.

Design

The board is compatible with V12 and V15 robots, but NOT V11s. This is due to header differences between V12 and V11. The J1 header is for V12, the J2 header is for V15. J4 is the JTAG header. J5 is a general IO header provided with much consternation and a very late night so James could use the board to prototype some project. He will probably never use it for said project. J3 is for the servo.

Previous versions of the board could support up to three servos, two motors and a servo, or a stepper motor and a servo. Once the mechanical team determined only one servo was needed the design was greatly simplified for elegance and a lack of board space.



Assembly

Do not populate the switch or the J1 header as there are NO plans to attach grippers to v12s. Parts should be located in a bin labeled “Gripper Board Parts” in the soldering room. The BOM in the SVN for v3 of the gripper contains all needed information for parts.

NOTE: The hall effect sensor U4 shares a trace with a JTAG line. The hall effect sensors interferes with the signal, making the board non-programmable while the sensor is present. The solution is to cut the trace and solder a wire from the sensor to a different unused IO port as specified in the diagram.

Board-Side Software

The software is mainly interrupt driven with a main loop which is used to update and average the current sense values and update the message, get IO values and update the SPI message, and update the servo message.

The board and the robot communicate via SPI with the robot serving as master. The board will only transmit its message if it has received a sensible message from the robot. The message is transmitted in 7 bytes: 3 sync bytes (‘BOT’), 3 information bytes, and a checksum. The robot sends only one useful byte, the next position for the servo. The board sends back the actual servo position, the current sense average, and a byte which contains information about the Hall Effect sensors and whether or not the force exerted is “high”.

The servo is controlled with a PWM signal which had a period of 25 ms and PWM of 2% to 10% (500 us to 2500 us). This is the standard method of controlling a servo. The PWM length is calculated from a raw value sent over SPI by offsetting and scaling the value into the correct number of clock cycles. If current is high, the PWM signal will be calculated based off the previous value by moving away from the direction of force. This is because we prioritize exerting too much force over the desired final position. If current is low, the PWM signal will be calculated based on the incoming SPI message as well as the previous value. This is to prevent jerky movement. The servo will oscillate slightly while gripping in order to maintain the appropriate level of force. However, this is slight and I believe unnoticeable. The calculated servo value that is output to the servo is also sent over SPI so that the robot knows what position it is actually at.

The current sense signal is an analog value output by the chip. The built in ADC is used and is triggered off of a timer. A sample is taken every 25 ms (how often the servo is updated) as more samples would be superfluous. The average is calculated during the main loop and then packed over SPI.

GPIO is used to keep track of the soft (hall effect) stops and if current is considered high. The correct ports are polled for the soft stops. The current sense value is compared to thresholds to determine if the current is high. High and low thresholds are both used to ensure that the force is consistently low or high enough to change value. This information is packed into the third information byte sent over SPI.

Robot-side software

The robot receives and sends SPI messages in a similar-ish manner to the board, but this is not important to understand for the gripper code. Code for the gripper is in robotcode/ronelibs/roneLib/src/expansions. This code provides a framework for interacting with the gripper. There are functions to initialize communication from the gripper, set the servo value, read the current servo value, read the current sense value, check if force is high, and check if either soft stop had been activated.

TODO: add higher level functions to OPEN and CLOSE the gripper (CLOCKWISE and COUNTERCLOCKWISE may be preferred names). Create behavior to respond to soft soft (turn around in the reverse direction). Use force high and actual servo position to update the desired position (once we have gripper, we have no need to insist on gripper harder)