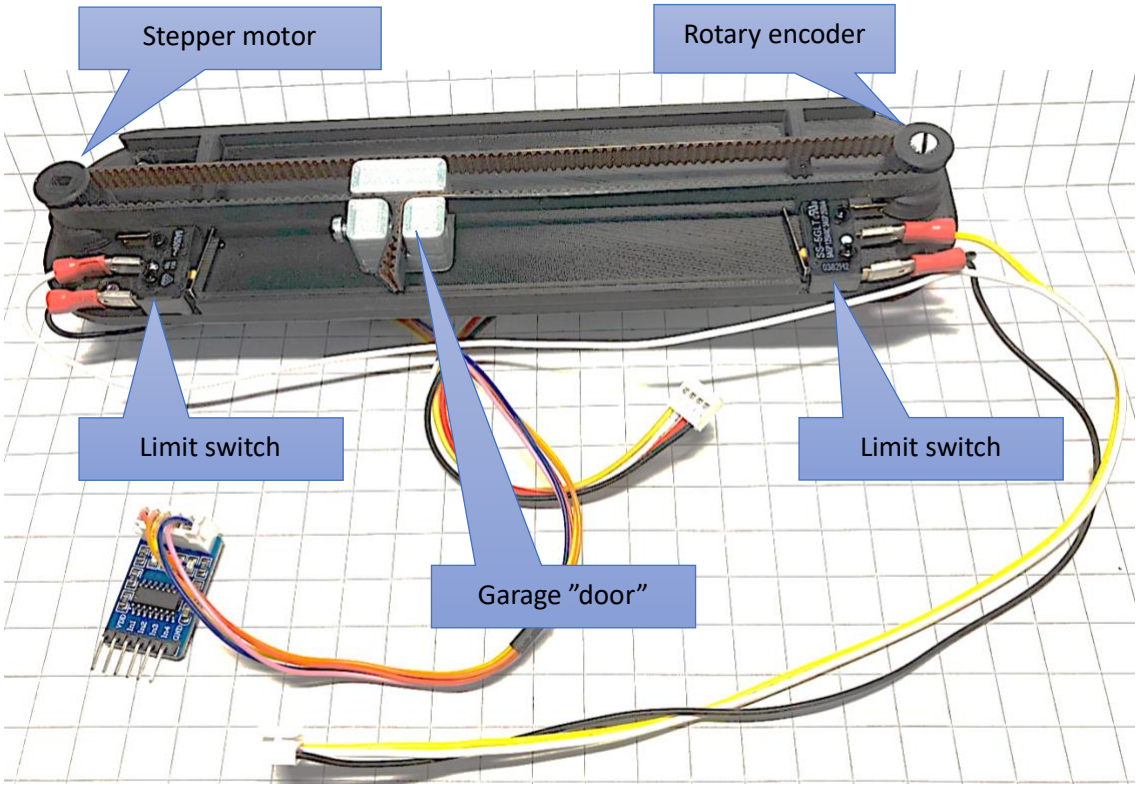
Garage door opener

Garage door opener consists of the following parts: a stepper motor to move the door up and down, a rotary encoder detect door movement and movement direction, two limit switches to detect when the door has reached the end of the movement range, and a door. The door in our test setup is a plastic block that is driven back and forth by the motor.



The door opener can be operated both locally and remotely. Buttons are used for local control and the controller subscribes to an MQTT topic to receive remote commands. The garage door opener communicates its state by publishing MQTT messages. The status indicates if the door is open or closed and if the controller is in an error state, for example the door is stuck.

Important information

Never turn the stepper motor or move the belt by hand!

Wiring and pin assignments

Test system has a 6-pin JST connector and a 4-pin Grove connector. Connect the JST connector to the 6-pin connector on the stepper driver board. There is only one 6-pin connector in the system so there is no risk of incorrect wiring. There are multiple grove connectors on the PCB. Connect the Grove connector to connector with two GPIOs.

Test system pin assignments:

- Stepper motor controller – GP2, GP3, GP6 and GP13
  - All four pins are outputs.
  - Pins are connected to the stepper motor driver pins IN1 – IN4
- LEDs
  - GP20 - GP22
- Rotary encoder
  - Requires grove connector with two GPIOs
  - Encoder has 20 detents per turn → it can detect turning in 18 degree steps
- Limit switches
  - Requires grove connector with two GPIOs
  - Limit switches require pull-up resistors

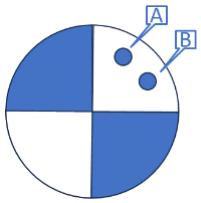
Rotary encoder

RP2040 has two GPIO banks that can trigger interrupts. Second GPIO bank is dedicated to external flash interface so user programs can use only GPIO bank 0. Each pin in the bank can be programmed to trigger an interrupt on HIGH/LOW level of rising or falling edge. It is possible to configure more than one trigger per pin, but the most common case is to configure only one type of trigger.

Study https://www.raspberrypi.com/documentation/pico-sdk/hardware.html#hardware_gpio. Pay attention to assigning callbacks to GPIO pins.

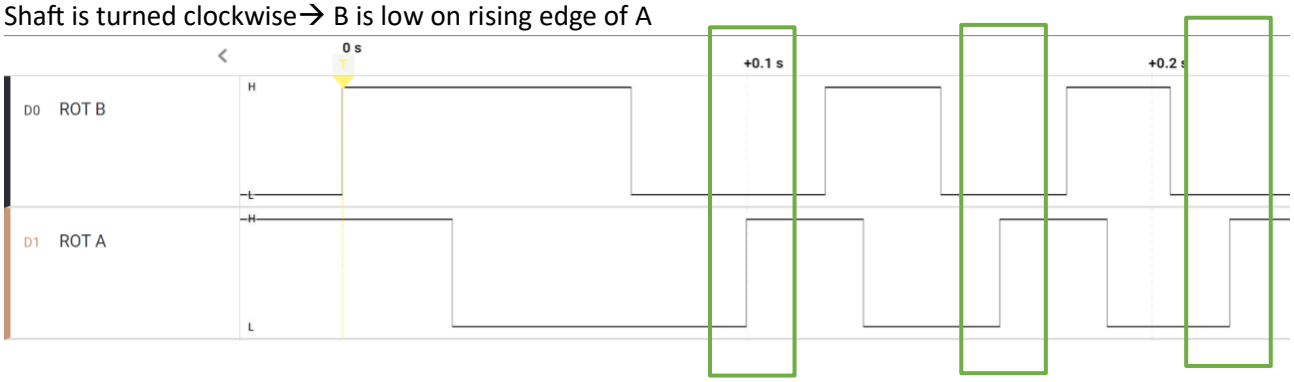
Uses queues to pass turn events from ISR to your progam. See: https://www.raspberrypi.com/documentation/pico-sdk/high_level.html#group_queue for queue API.

Rotary encoder has two outputs that change their state when the encoder is turned. The outputs are typically named A and B. The encoder is built so that only one of the outputs changes at a time when the shaft is turned. Depending on the rotation either A or B makes contact first. To decode direction of turn we can choose one of the signals as clock and monitor the change of the other on one edge of the clock.

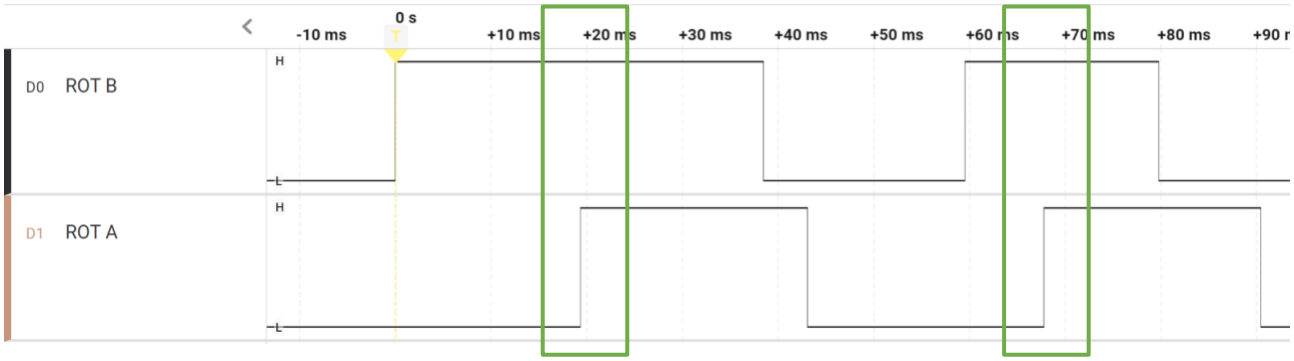


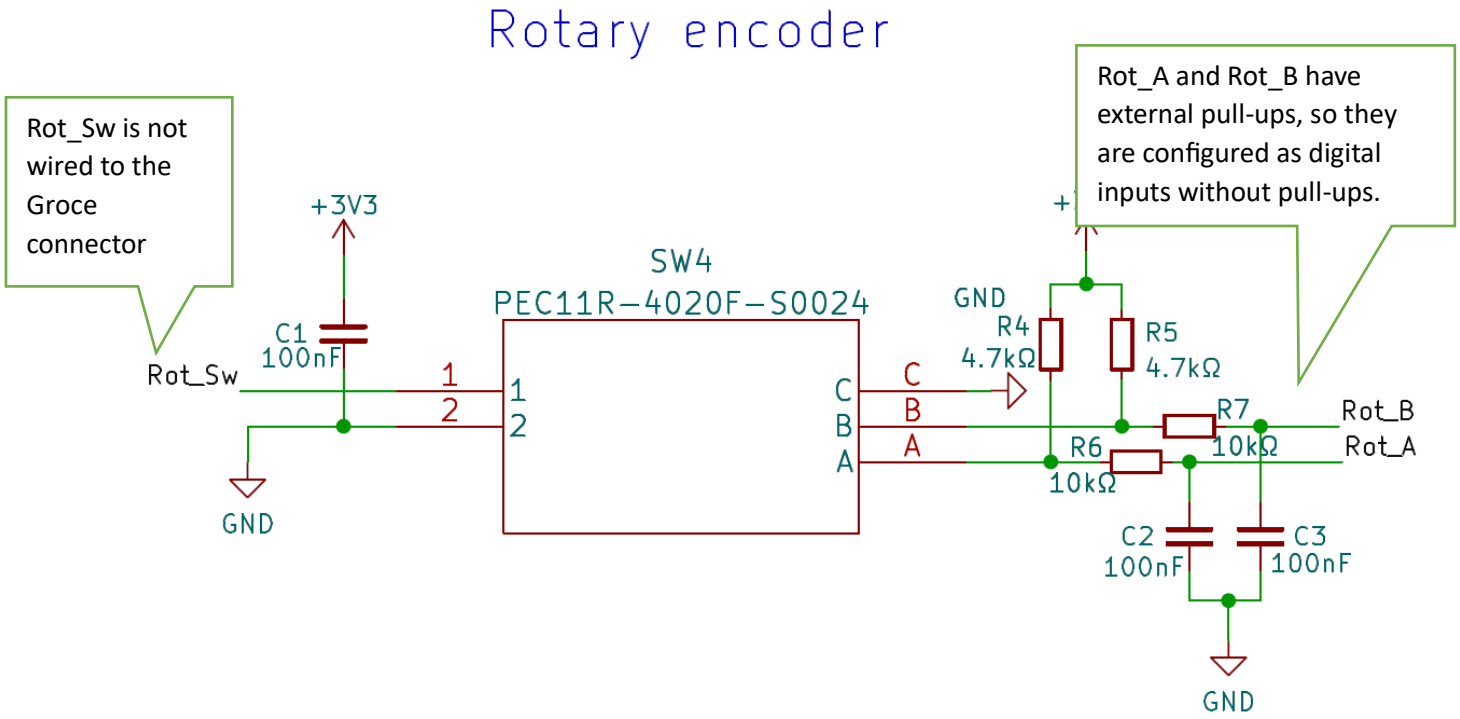
For example, if we choose A as the clock, we check the state of B on every rising edge of A. To ensure that we don't miss any of the edges we'll configure an interrupt on the rising edge of the pin A is wired to.

Shaft is turned clockwise→ B is low on rising edge of A



Shaft is turned counter-clockwise: B is high on rising edge of A





Project specifications

Door opener must have the following functionality:

- Calibration: Pressing SW0 and SW2 at the same time starts calibration. During calibration the door is run up and down and the number of steps between up and down position is determined using the limit switches and the rotary encoder to detect when the door stops at both ends. It is safe to run the motor against the limit switch body during calibration, but a properly implemented program should stop the motor when the switch is closed but the block does not yet hit the body of the switch. After calibration the door may not hit the body of either limit switch during normal operation.
- Status reporting: Program reports the following status through MQTT
    - Door state: Open, Closed, In between
    - Error state: Normal, Door stuck
    - Calibration state: Calibrated/Not calibrated
- Local operation: Pressing SW1 has following functionality:
    - Door is closed → door starts to open
    - Door open → door starts to close
    - Door is current opening or closing → door stops
    - Door was earlier stopped by pressing the button → door starts movement to the opposite direction (stopped during opening→close and vice versa)
- If the door gets stuck during opening/closing the motor is stopped, error state is reported, and door goes to not calibrated state.
- Opening button may not work in the non-calibrated state.
- Remote control may not work int the non-calibrated state.
- Status must be indicated locally using LEDs. Error state must be indicated by blinking an LED.

Minimum requirements

The door can be operated locally, and calibration rules work as described earlier. Object orientation is used in some parts of the program. Meeting minimum requirements gives you 50 points.

Advanced requirements

Program is implemented in object-oriented style.

Program preserves calibration and door state across power down or reboot.

Objects have clear responsibilities and reflect entities in the application/hardware.

Program sends status messages to an MQTT-server using topic “garage/door/status”.

Program subscribes to topic “garage/door/command” and executes the received commands. Command responses are sent to topic “garage/door/response”. Response must contain the command result if command was successfully executed or an error message if command can’t be completed. Note that status of the door must also be reported to “garage/door/status” if the command changes door status. User must be able to perform same operations through remote commands as with the local (button) interface.

Additional remote features count towards higher grade provided that the requirements stated above are met.