Control (control.cpp)

checkStopCondition()

SW_START.state —FALSE→ *start button not pressed* | state = COUNTING / vfd_setState(false)

else if

lengthDiff >= 0 —true→ *target reached, or already at target* | state = TARGET_REACHED / vfd_setState(false)

setDynSpeedLvl(uint8 lvlMax)

lengthRemaining
- < 50 → vfd_setSpeedLevel (0)
- < 200 → vfd_setSpeedLevel (1)
- < 500 → vfd_setSpeedLevel (2)
- else → vfd_setSpeedLevel (3)

limit to lvlMax

---

start counting (no motor)
(pull and attach wire manually to spool)

main loop
COUNTING → motor_stop()

state: COUNTING

SW_START.risingEdge

vfd_setSpeedLevel(1) / vfd_setState(true) / beep(2)

start or resume slow for some time (time for verifying it spools correctly)

state: WINDING_START

main loop
WINDING_START → checkStopCondition() → setDynSpeedLvl(1)

limit speed lvl to 1 can be slower if close though

2s

beep(1)

wind at dynamic speed (slower speed when closer to target)

state: WINDING

main loop
WINDING → checkStopCondition() → setDynSpeedLvl()

state: TARGET_REACHED

todo:
- reverse when target exceeded?

!autoCutEnabled && lengthRemaining > 0

autoCutEnabled == true

todo:require reset switch to be able to start again?

state: AUTO_CUT_WAITING

main loop
AUTO_CUT_WAITING → indicate cut pending on display / beep countdown / blink lamp

SW_RESET / SW_CUT / !SW_AUTO_CUT

delay passed

cutter_start()

state: CUTTING

todo:
- ability to CANCEL cutting
- e.g. cancel when buttons released?

cutter_isRunning() == false

lengthNow = 0

---

main loop
lengthRemaining = lengthTarget - lengthNow

main loop
SW_PRESET2 && (SW_PRESET1 || SW_PRESET3) && state != MANUAL

switch to manual mode at any state

state: MANUAL

manually control the motor via preset buttons and potentiometer

—true→

SW_P2 == false

else if

SW_P1 → vfd_setSpeedLevel(POTENTIOMETER) / vfd_setState(true, REV)

else if

SW_P3 → vfd_setSpeedLevel(POTENTIOMETER) / vfd_setState(true, FWD)

Wire cutter (cutter.cpp)

main loop

cutter_handle()

cutter_stop()

state:
IDLE

state:
CANCELED

cutter_start()

cutter_start()

state:
TIMEOUT

GPIO_RELAY on
timestamp_turnedOn = now()

cutter_start()

motor running but
still at start position

state:
START

checkTimeout()

SW_CUTTER == LOW

not at idle position
(switch closed / not switched to gnd)

motor running but
still at start position

state:
CUTTING

checkTimeout()

SW_CUTTER == HIGH

at idle position
(switch open / not switched to gnd)

GPIO_RELAY off

---

cutter_start()

GPIO_RELAY on
timestamp_turnedOn = now()
state = START

---

cutter_stop()

GPIO_RELAY off
state = CANCELED

---

cutterState_t
cutter_getState()

return state

---

bool cutter_isRunning()

state

return false ←else

state == START
|| state == CUTTING

return true

---

bool checkTimeout()

now() -
timestamp_turnedOn
.

> msTimeout

GPIO_RELAY = OFF
timestamp_turnedOff = now()
state = TIMEOUT
beep (5)
return true;

else

return false

Features/Ideas - List:
- slow down when close to target
- stop when no movement recognized
- manual reverse control? e.g. unwind again

---

rotary encoder (control.cpp)

process rotary
encoder using
interrupts (library)

Interrupt
A/B encoder

process
(rotary_encoder lib)

send event to queue

calculate distance
each cycle

main loop

encoder_getState

lengthNow = state.position /
600 * ROLL_DIAMETER * PI
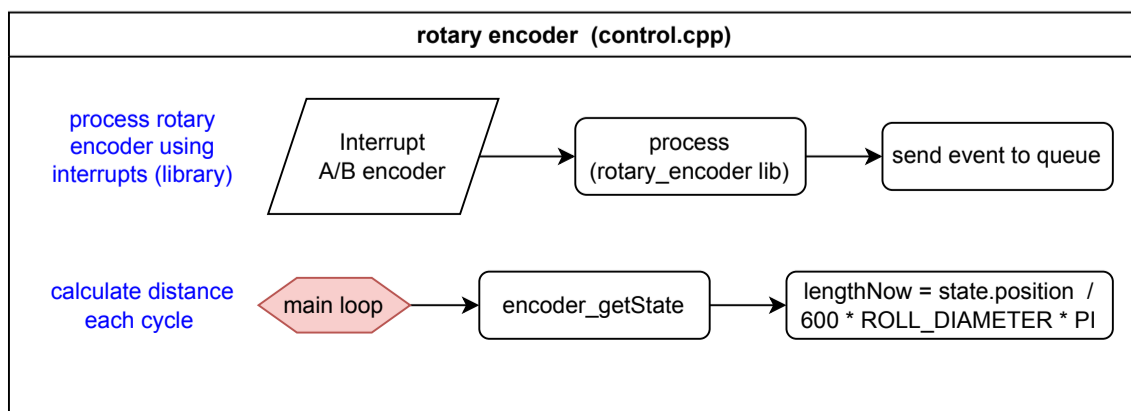
Input

- Button Start
- Button Reset
- Button Set
- Potentiometer

- Preset 1-3

Output

- 2x 8 digit 7-segment display
- buzzer

- VFD
    - D0 - D2
    - FWD
    - REV

- Control lamp

Button actions (control.cpp)

main loop

reset counter with
reset button

set custom target length using
potentiometer and set button

todo:
- minimum target length
- at zero entire program will -> displays
msg and cuts

SW_RESET.state

true

// lengthNow = 0
encoder_reset()

SW_SET.state

true

voltage = adc_read(gpio_poti)
lengthTarget = voltage * x

set preset length 1

set preset length 2

set preset length 3

SW_PRESET1.state

SW_PRESET2.state

SW_PRESET3.state

true

true

true

lengthTarget = 3000

lengthTarget = 5000

lengthTarget = 10000

Display

TODO: this needs to
be updated with new
display class

show welcome
message on startup
(before while loop)

startup

drawText ("name, date")
delay(1000)
for {
shift array "hello" 1 place every 50ms
drawText(array)
}

main loop

buf_disp1 = lengthNow

buf_disp2 = lengthTarget

show target length by
default

SW_SET.state —true→

TODO BLINK TARGET (clear
every 400ms for 100ms
buf_disp2 = " "

if setting target via poti
blink target length

//buf = sprintf("%s%s", buf_disp1, buf_disp2)
//drawText(buf)

drawText(0, buf_disp1)
drawText(8, buf_disp2)

draw both buffers to
display