

### UART board communication

Planning data / structs need to be sent between the two boards here:

**board control**

- joystick
- encoder, switch
- display

**MOTORCTL\_STATUS**  
[ interval, slow ]

timestamp  
motorcommands (current actual duty, direction)  
motor currents  
axle rotating speed  
temperatures  
- motors  
- drivers  
- brake resistors  
fan target state [ ] handled by motorctl board  
battery voltage  
config?

**board motorctl**

- 2x driver
- 2x currentsensor
- 2x light barrier axle speed
- 2x brake resistor
- temp sensors
- motor
- driver
- brake resistor
- fan

UART

**MOTOR COMMANDS**  
[ interval, fast ]

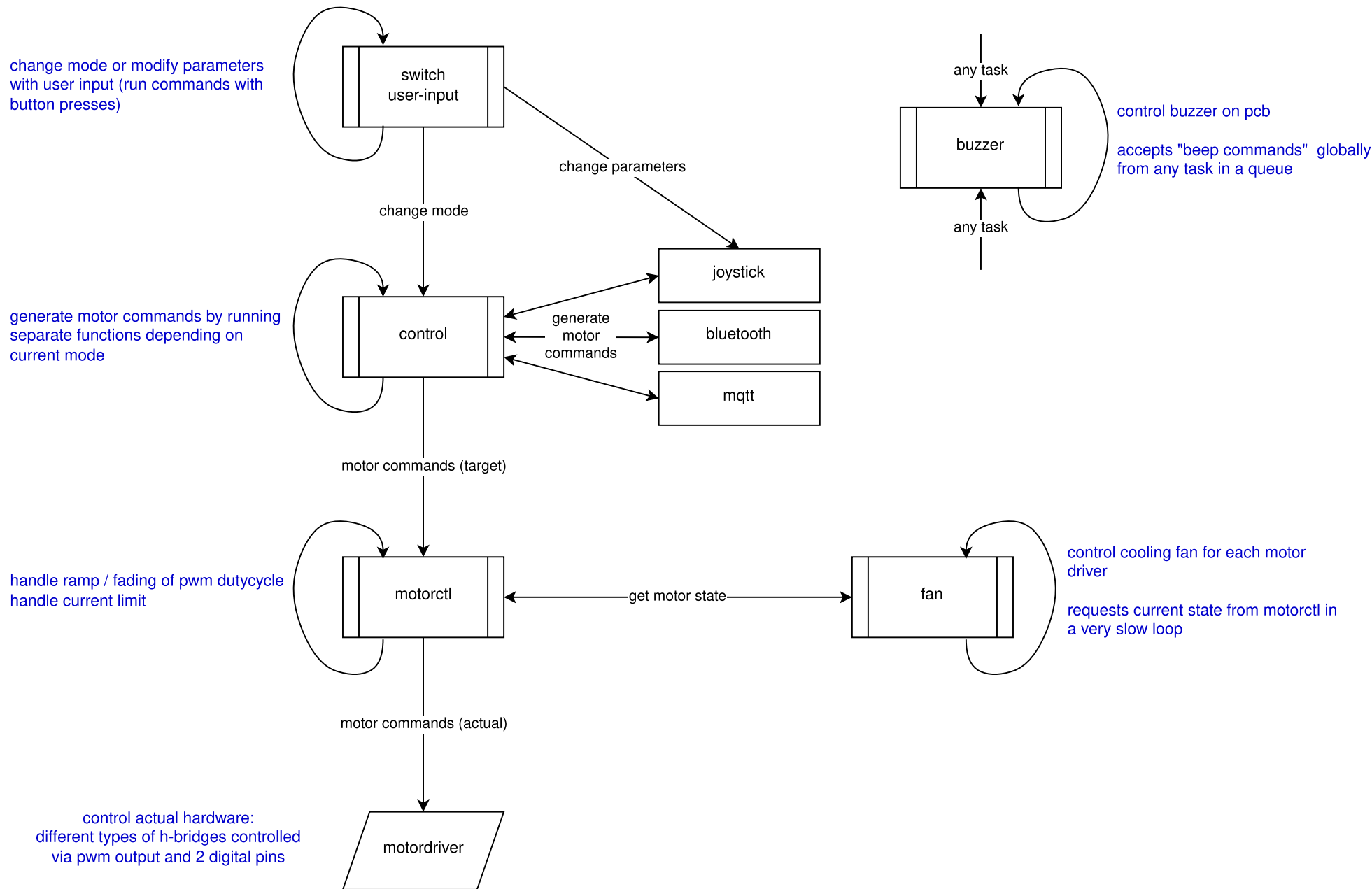
timestamp  
motorcommands  
- motorCommand\_t left  
- motorstate  
- duty

**CONFIG**  
[ manual / on change ]

timestamp  
motorctl\_config\_t  
- msFadeAccel  
- msFadeDecel  
- bool currentLimitEnabled  
- (currentSensor\_adc)  
- (currentSensor\_ratedCurrent)  
- currentMax  
- deadTimeMs

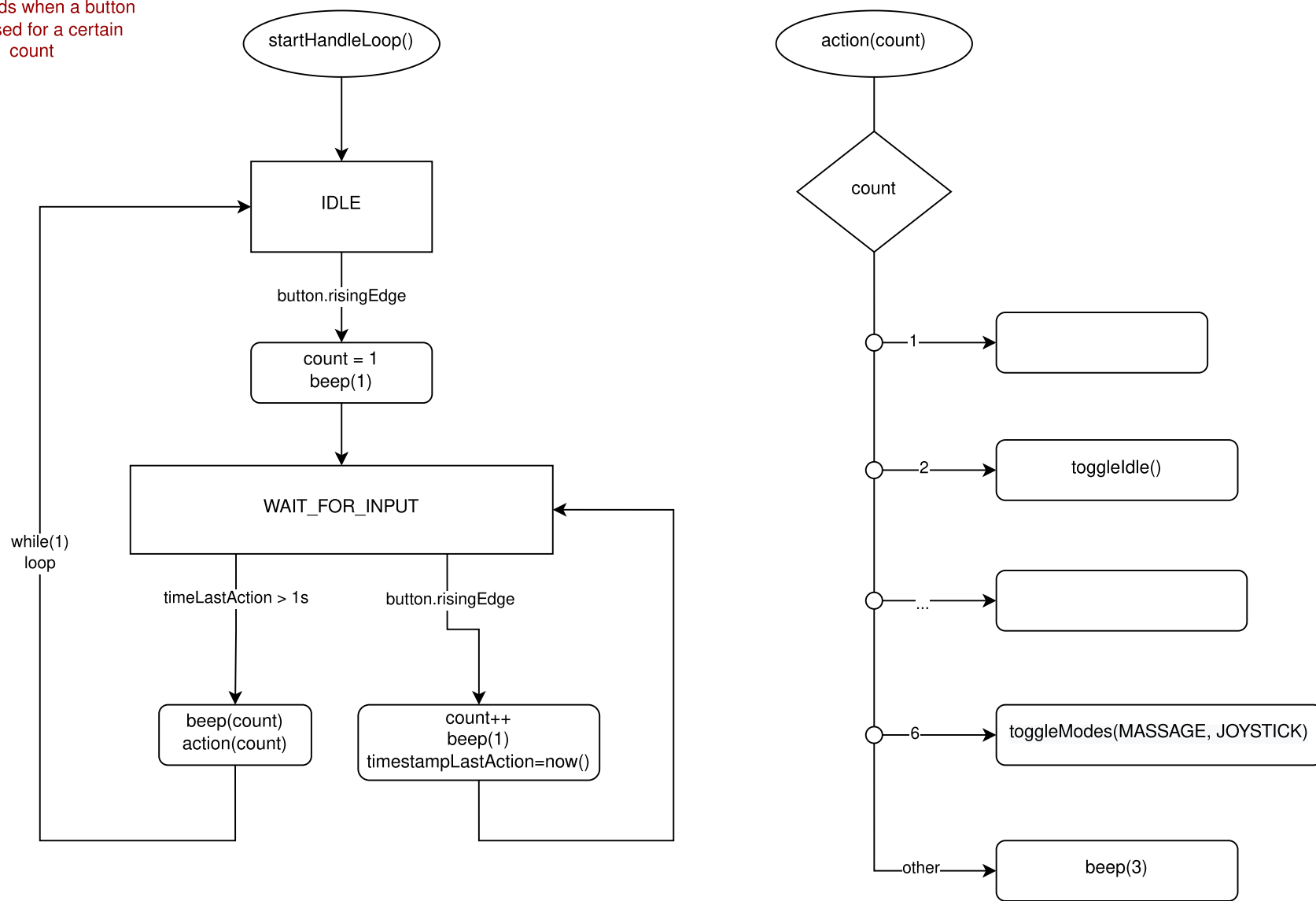
**NOTE: the below diagrams might be significantly outdated -- needs update to match the actual code again**

### Tasks / Threads overview



### class buttonCommands (button.hpp, button.cpp)

class which makes it possible to run different commands when a button is pressed for a certain count



class controlledArmchair (control.hpp, control.cpp)

run functions corresponding to the current mode that generate motor commands, send the resulting commands to motorctl

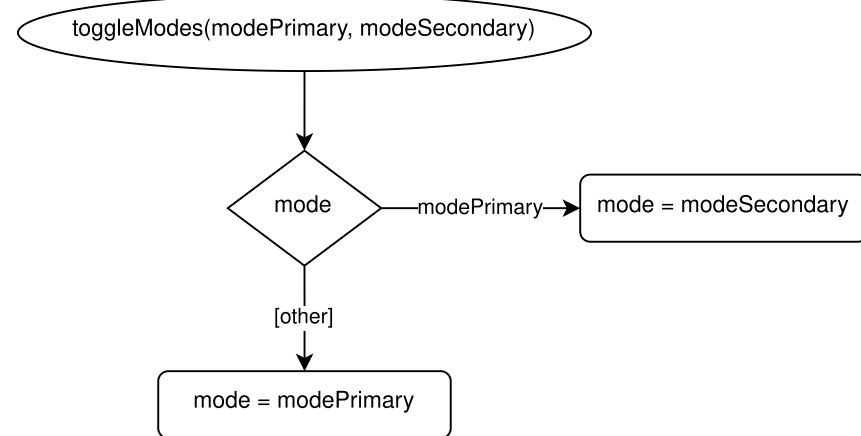
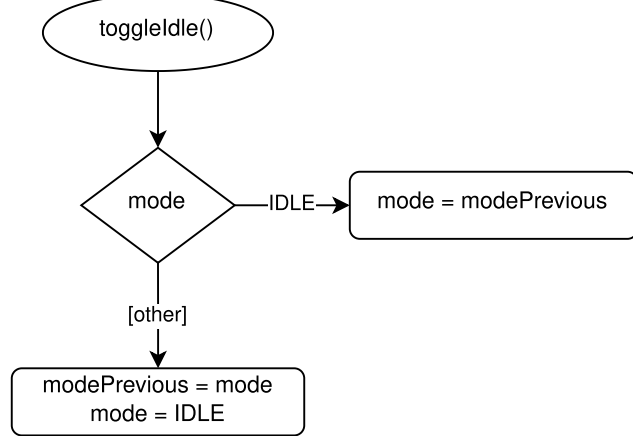
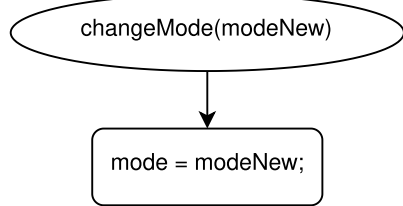
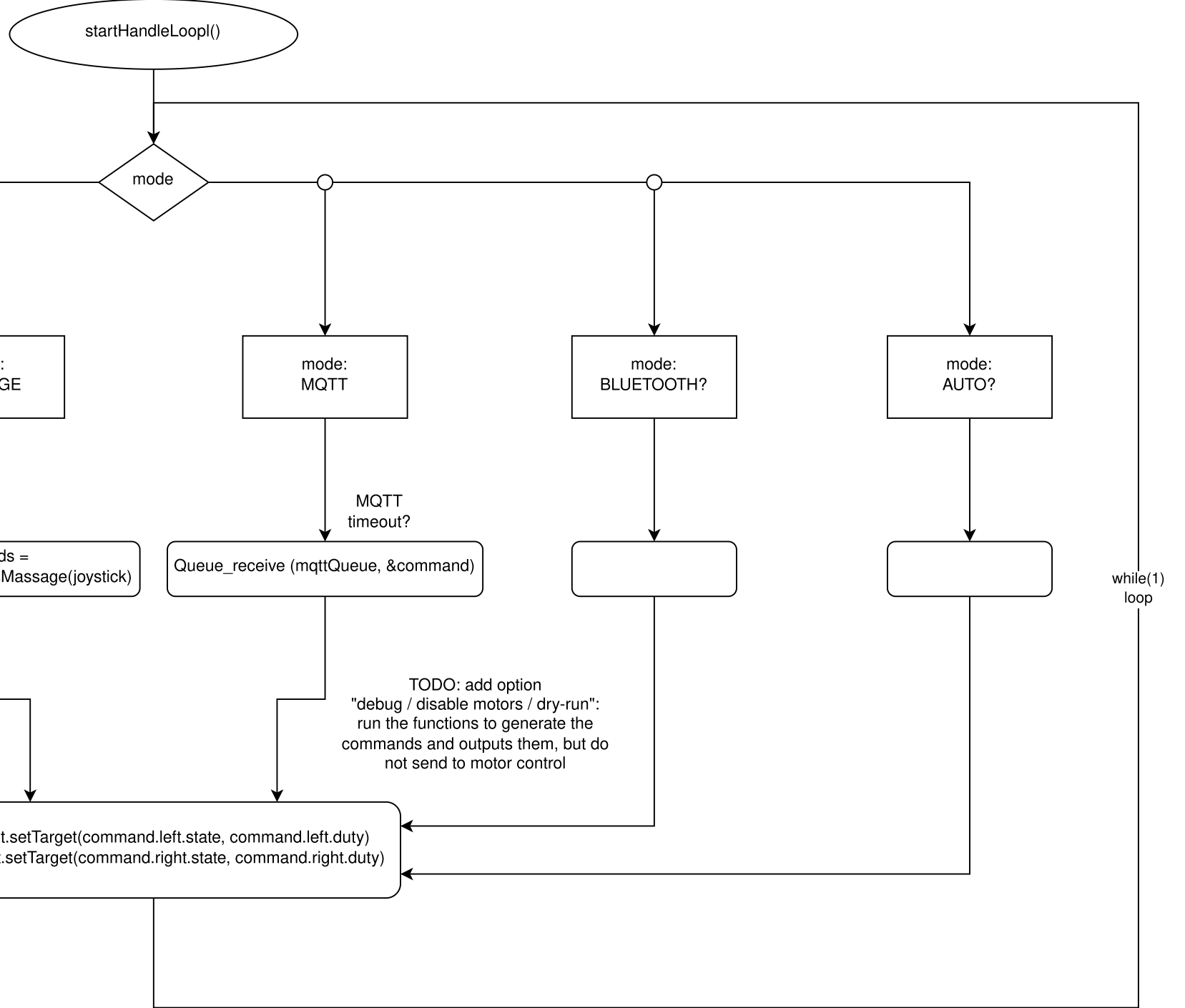
generate motor commands by calling functions corresponding to current mode

send target state and duty to controlledMotor object of each motor (controlledMotor applies ramp and current limit and commands motor driver repeatedly)

switch to specific mode

toggle between IDLE and previous or default mode

toggle between two specific modes e.g. button press switches between MESSAGE and JOYSTICK

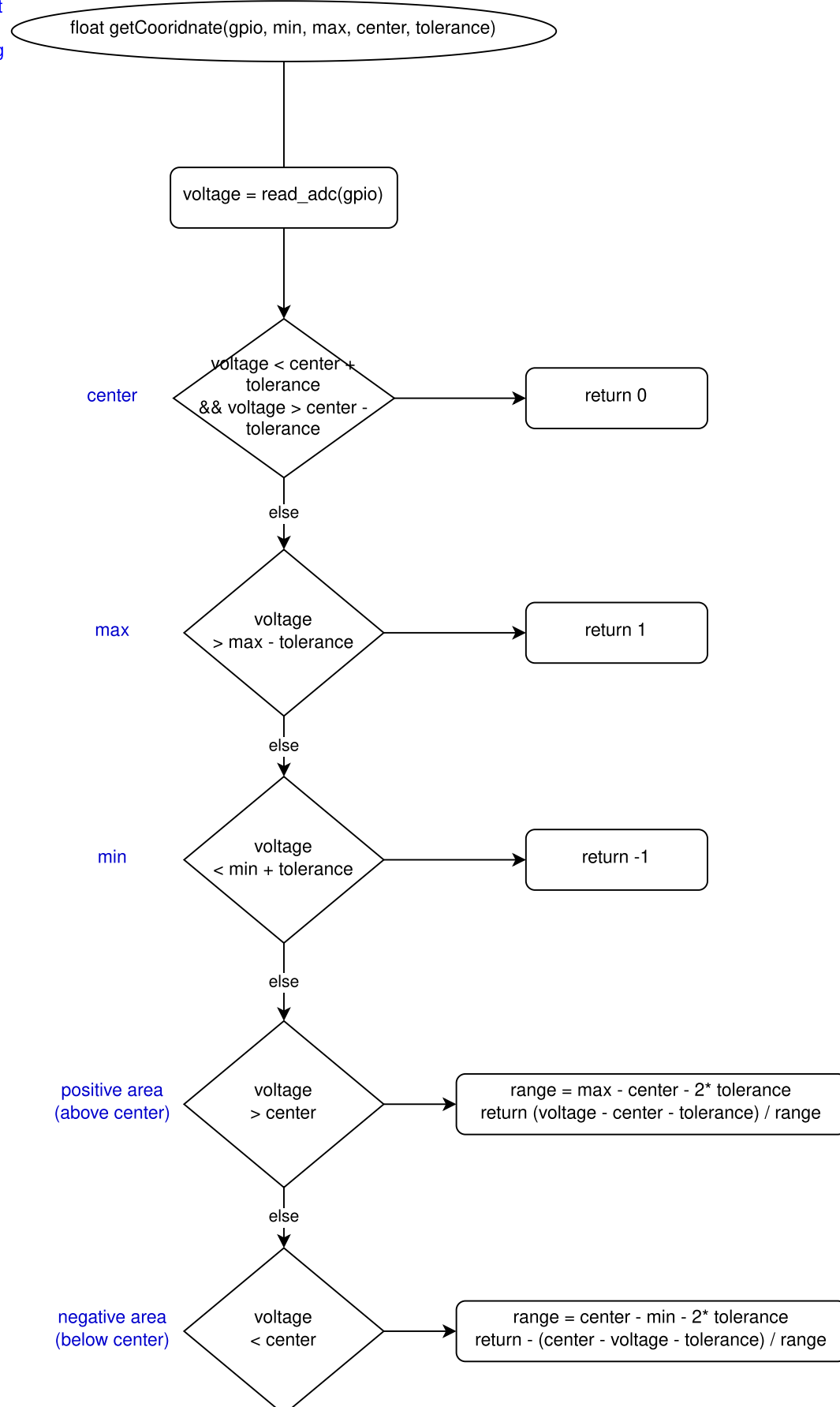
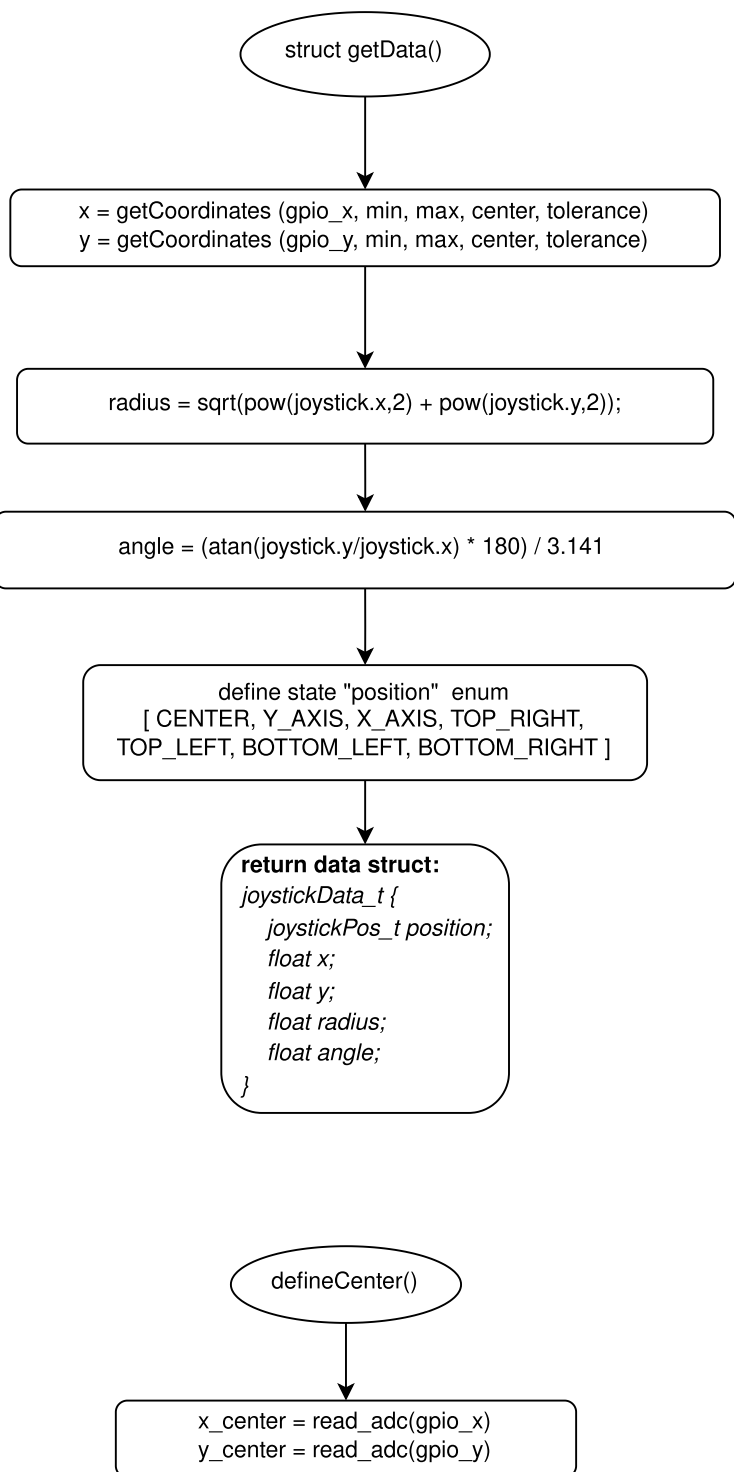


class evaluatedJoystick (joystick.hpp, joystick.cpp)

function that makes it possible to get a struct with current state and data of the joystick.

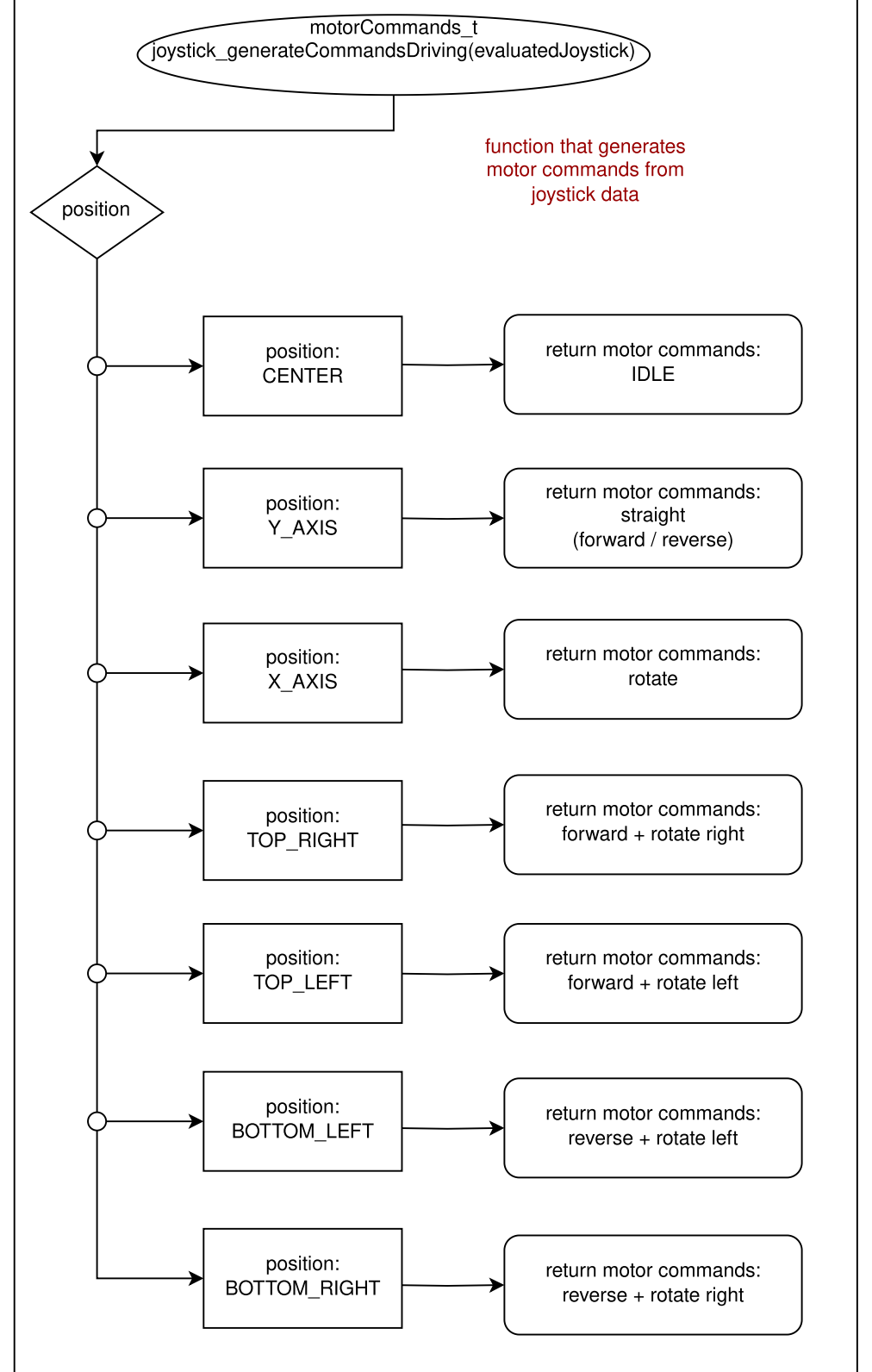
this can be used as input in other functions or tasks

private function that reads an analog input and calculates coordinates according to given parameters



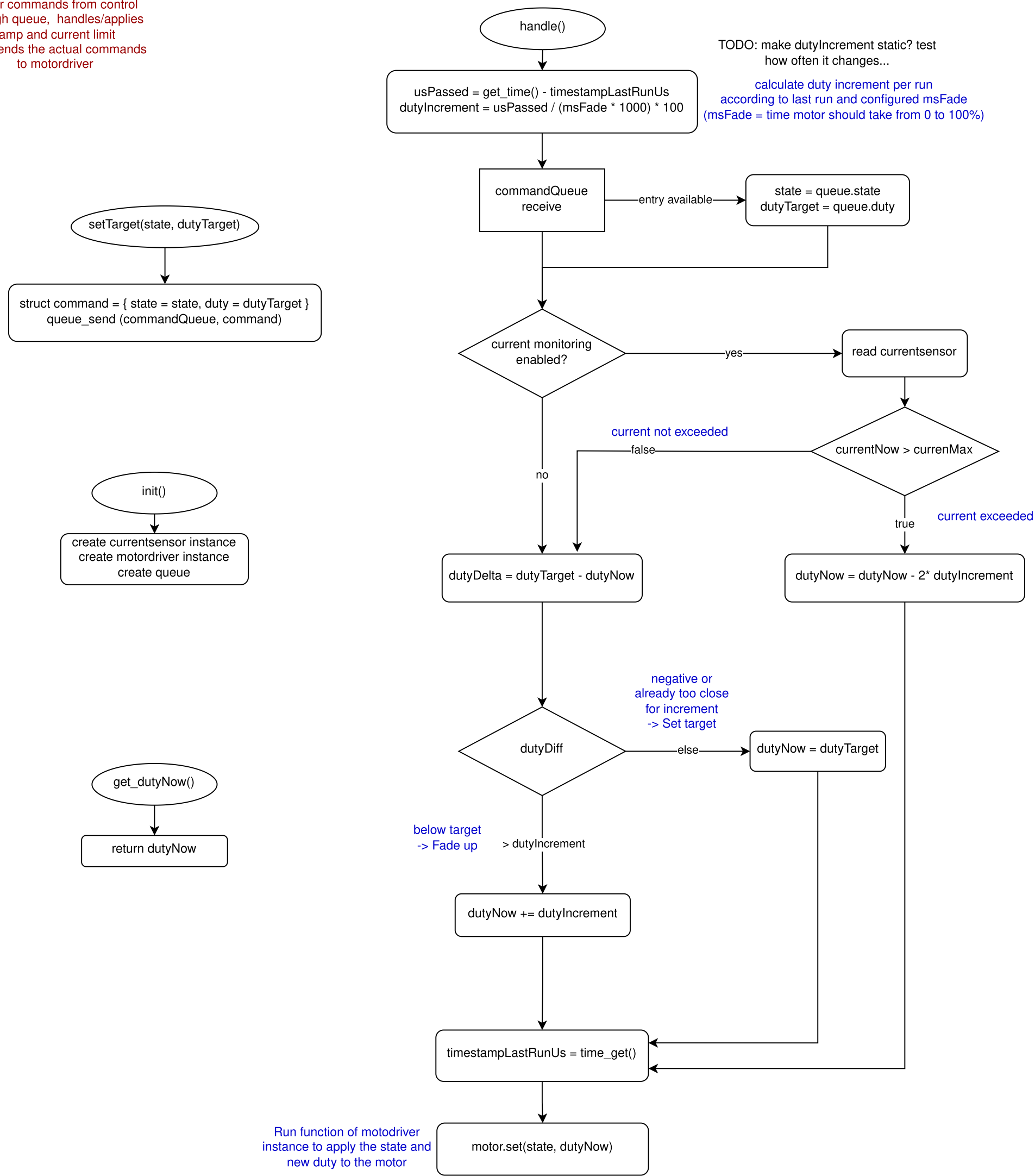
function: joystick\_generateCommandsDriving

function that generates motor commands from joystick data



**class controlledMotor (motorctl.hpp, motorctl.cpp)**

class for each motor that receives motor commands from control through queue, handles/applies ramp and current limit then sends the actual commands to motordriver



**class currentsensor**

