#define xQueuePushBackFromISR(queue, value) auto ___value = value; if(uxQueueSpacesAvailable(queue) == 0){ auto ___temp_value = ___value; xQueueReceiveFromISR(queue, &___temp_value, 0); Serial.print("pop |");} xQueueSendToBackFromISR(queue, &___value, &pFALSE) #define xQueuePushBack(queue, value) auto ___value = value; if(uxQueueSpacesAvailable(queue) == 0){ auto ___temp_value = ___value; xQueueReceive(queue, &___temp_value, 0); } xQueueSendToBack(queue, &___value, 0) #define xQueuePushBack(queue, value) auto ___value = value; if(uxQueueSpacesAvailable(queue) == 0){ auto ___temp_value = ___value; xQueueReceive(queue, &___temp_value, 0); } xQueueSendToBack(queue, &___value, 0) #define WRAP(classname, funcname) static funcname(void* obj){ ((classname*) obj)->##**funcname(); }; void** ##funcname()

/ `xQueuePushBack(queue, value)` pushes a value to a queue. if the queue is full, it automatically removes the first value in the queue `xQueuePushBackFromISR()` is basically the same thing `WRAP(class, func)` wraps a member function `func` from `class`, so it can be used as a task (note: tasks and freeRTOS stuff only use C functions so C++ functions need to be wrapped. also, the wrapped task takes `(void*)this` as an argument) /

class Sensor{

protected: QueueHandle_t queue;

public: int pin; void begin(int pin, int priority=1, int memory=1024){

```
this->queue = xQueueCreate(3, sizeof(int));
this->pin = pin;

char buf[25];
sprintf(buf, "SENSOR TASK (pin %d)", pin);

xTaskCreate(
    this->task,
    buf,
    memory,
    this, // need to pass `this` as an argument
    priority,
    NULL
);
```

```
}

void WRAP(Sensor, task){  // note: `task` is  the name of the function (being wrap
                          // functions need to be wrapped because freeRTOS cant ha
                          // also dont define `__task` as a function, it gets used

    while(true){

        xQueuePushBack(
            this->queue,
            analogRead(this->pin)
        );

        vTaskDelay(100);
    }
}

int read(int ticks=0){
    int value = -1;

    xQueueReceive(this->queue, &value, ticks);

    return value;
}

};
```