```c
#include "main.h"

void motorSlewTask(void *parameter){
    int motorIndex;
    int motorPort;
    int motorTmp;

    mutexTake(motorReqMutex, 100);
    for(motorIndex=0;motorIndex<MOTOR_NUM;motorIndex++)
    {
        motorReq[motorIndex] = 0;
        motorSlew[motorIndex] = MOTOR_DEFAULT_SLEW_RATE;
    }
    mutexGive(motorReqMutex);

    while( true )
    {
        int requestCopy[10];

        mutexTake(motorReqMutex, 100);
        for(int i = 0; i < 10; i++){
            requestCopy[i] = motorReq[i];
        }
        mutexGive(motorReqMutex);
        for( motorIndex=0; motorIndex<MOTOR_NUM; motorIndex++)
        {
            motorPort = motorIndex + 1;
            mutexTake(motorMutexes[motorIndex], 100);
            motorTmp = motorGet(motorPort);
            mutexGive(motorMutexes[motorIndex]);
            if( motorTmp != requestCopy[motorIndex] )
            {

                if( requestCopy[motorIndex] > motorTmp )
                {

                    motorTmp += motorSlew[motorIndex];

                    if( motorTmp > requestCopy[motorIndex] )
                    motorTmp = requestCopy[motorIndex];
                }

                if( requestCopy[motorIndex] < motorTmp )
                {
                    motorTmp -= motorSlew[motorIndex];

                    if( motorTmp < requestCopy[motorIndex] )
                    motorTmp = requestCopy[motorIndex];
                }

                mutexTake(motorMutexes[motorIndex], 100);
                motorSet(motorPort, motorTmp);
                mutexGive(motorMutexes[motorIndex]);
            }
        }
        delay( MOTOR_TASK_DELAY );
    }
}

void waitForTasks(){
    bool finger = true;
    bool lift = true;
    bool wheels = true;

    while(finger == true || wheels == true || lift == true){
        mutexTake(runFingerMutex, 100);
        finger = runFinger;
        //printf("runFinger = %d\n", runFinger);
        mutexGive(runFingerMutex);

        mutexTake(runLiftMutex, 100);
        lift = runLift;
        //printf("runLift = %d\n", runLift);
        mutexGive(runLiftMutex);


        mutexTake(runWheelsMutex, 100);
        wheels = runWheels;
        //printf("runWheels = %d\n\n", runWheels);
        mutexGive(runWheelsMutex);

        delay(20);
    }
}

void stopAllMotors(){
    stopDrive();
    stopLift();

    mutexTake(motorMutexes[fingerY - 1], 100);
    motorStop(fingerY);
    mutexGive(motorMutexes[fingerY - 1]);
}

void zeroDriveSensors(){
    encoderReset(leftQuad);
    encoderReset(rightQuad);
}

void zeroAllSensors(){
    zeroDriveSensors();
    encoderReset(liftQuad);
}

int programSelected(int segments){
    int oneValue = clamp(analogRead(potOne)/(4095 / segments),0,segments - 1);
    int twoValue = clamp(analogRead(potTwo)/(4095 / segments),0,segments - 1);
    return oneValue + twoValue;
}

int clamp(int var, int min, int max){
    if(var > max){
        return max;
    }else if( var < min){
        return min;
    }else{
        return var;
    }
}
```