**//fight-control**

起飞函数：

bool monitoredTakeoff(DJI::OSDK::Vehicle\* vehiclePtr, int timeout = 1);

降落函数：

bool monitoredLanding(DJI::OSDK::Vehicle\* vehiclePtr, int timeout = 1);

指定方向距离：

位置控制。允许设置当前位置的偏移量。飞机将移动到该位置并停留在那里。典型的用途是作为外环中的构建块，不需要很多快速变化，可能是几个点的轨迹。为了更平滑的过渡和响应，应该将轨迹转换为姿态设定点并使用姿态控制或转换为速度设定点并使用速度控制。

bool moveByPositionOffset(DJI::OSDK::Vehicle \*vehicle, float xOffsetDesired,

float yOffsetDesired, float zOffsetDesired,

float yawDesired, float posThresholdInM = 0.2,

float yawThresholdInDeg = 1.0);//传参顺序：x轴偏移量、y轴偏移量、z轴偏移量、机头旋转角度

//计算两对GPS坐标之间的局部NED偏移

void localOffsetFromGpsOffset(DJI::OSDK::Vehicle\* vehicle,

DJI::OSDK::Telemetry::Vector3f& deltaNed,

void\* target, void\* origin);//deltaNed中保存计算后的偏移量

**//telemetry**

//可以返回一个遥测数据，多加一个参数

获得遥测信息

bool getBroadcastData(DJI::OSDK::Vehicle\* vehicle, int responseTimeout = 1, &telData);//M100仅有此函数，telData中保存了返回的遥测数据类

**//carmera-gimbal**

//设置云台/相机角度，默认的持续时间为2秒，以增量形式改变

bool doSetGimbalAngle(Vehicle\* vehicle,int x,int y,int z,int Duration=20, int isAbsolute=0);

//显示当前云台角度

void displayResult(RotationAngle\* currentAngle);

//相机控制用，于实现拍照和摄影分开控制,默认为摄影模式，摄影时间为6秒。shootPhoto=0（摄影模式）shootPhoto=1（拍照模式）

bool gimbalCameraControl(Vehicle\* vehicle,int shootPhoto=0,int time=6);

**//mission**

沿多边形飞行：以当前GPS位置生成GPS航点多边形，执行起飞然后运行航点任务。任务的某些方面（速度，航向）可通过DJI Go应用程序控制。

bool runWaypointMission(DJI::OSDK::Vehicle\* vehicle, uint8\_t numWaypoints,

int responseTimeout); //传参顺序：Vehicle、多边形边的数量、回应超时

绕点沿圆飞行：在当前的GPS位置周围设置一个Hotpoint（Point of Interest）任务，执行起飞然后执行任务。任务的许多方面（速度，航向，半径，运动方向）都可以通过DJI Go应用程序控制。

bool runHotpointMission(DJI::OSDK::Vehicle\* vehicle, int initialRadius,

int responseTimeout); //传参顺序：Vehicle、圆的半径、回应超时

通用信息：

class LinuxSetup

{

public:

void setupEnvironment(int argc, char\*\* argv);

void initVehicle();

bool validateSerialPort();

}