```
#include <iostream>
#include <opencv2/opencv.hpp>
#include "depthai/depthai.hpp"



//Using DepthAI in order to detect a person. Get their depth information, and also send
coordinates.
int depthAI() {
    using namespace std;
    using namespace dai;

    // Create pipeline type from depthAI libary. A pipeline is a class that idenfitifes how
    // the camera, neural network, and depth nodes connect.
    dai::Pipeline p;

    // Accessing camera
    auto camRgb = p.create<node::ColorCamera>();
    camRgb->setPreviewSize(100,100);//Unsure what this size should be
    camRgb->setInterleaved(false);
    camRgb->setColorOrder(ColorCameraProperties::ColorOrder::BGR);
    camRgb->setFps(30);

    // Acessing Stereo for depth
    auto monoLeft = p.create<node::MonoCamera>();
    auto monoRight = p.create<node::MonoCamera>();
    auto stereo = p.create<node::StereoDepth>();

    //Setting left side of stereo cameras
    monoLeft->setBoardSocket(CameraBoardSocket::LEFT);
    monoLeft->setResolution(MonoCameraProperties::SensorResolution::THE_400_P);

    //Right camera of stereo cameras
    monoRight->setBoardSocket(CameraBoardSocket::RIGHT);
    monoRight->setResolution(MonoCameraProperties::SensorResolution::THE_400_P);


    stereo->setDefaultProfilePreset(node::StereoDepth::PresetMode::HIGH_DENSITY);
    stereo->setLeftRightCheck(true);

    //Setting mono camera outputs into stereo outputs
    monoLeft->out.link(stereo->left);
    monoRight->out.link(stereo->right);
```

```
    // Creating Spatial Detection Network for coordinates and mapping. Using mobilenet model
    // which is the general purpose model used by depthAI
    auto spatialDet = p.create<node::MobileNetSpatialDetectionNetwork>();
    spatialDet->setBlobPath("mobilenet-ssd_openvino_2021.4_6shave.blob");

    spatialDet->setConfidenceThreshold(0.5f);
    spatialDet->input.setBlocking(false);

    camRgb->preview.link(spatialDet->input);
    stereo->depth.link(spatialDet->inputDepth);

    // Configure depth boundaries(in mm)
    spatialDet->setDepthLowerThreshold(1000);   //1m (3.2ft)
    spatialDet->setDepthUpperThreshold(10000); // 10m (32ft)

    //outputing our detections/human detection
    auto xoutDet = p.create<node::XLinkOut>();
    xoutDet->setStreamName("detections");
    spatialDet->out.link(xoutDet->input);

    //Depth output
    auto xoutDepth = p.create<node::XLinkOut>();
    xoutDepth->setStreamName("depth");
    stereo->depth.link(xoutDepth->input);

    // Initilizing device
    dai::Device d(p);  // Unsure how to initialize a device as of rn
    auto qRgb = d.getOutputQueue("rgb", 8, false);  //Used to get the image frame
    auto qDet = d.getOutputQueue("detections", 8, false);  //Used to get detection objects
    auto qDepth = d.getOutputQueue("depth", 8, false); // Used to get depth of our detections


    // Main loop used to send depth information using cv to idenftify the person
    while(true) {
        if(det.label == 15){ //mobilenet-ssd_openvino_2021.4_6shave.blob supports multiple
classes of detections.
        //detection #15 is the class of a person.

        //Getting the frame of our camera
        auto inRgb = qRgb->get<ImgFrame>();
```

```
        auto frame = inRgb->getCvFrame();
        auto detections = qDet->get<SpatialImgDetections>();

        //Framing/getting dimensions of our detection
        for(auto& det : detections->detections) {
            int x1 = det.xmin * frame.cols;
            int y1 = det.ymin * frame.rows;
            int x2 = det.xmax * frame.cols;
            int y2 = det.ymax * frame.rows;

            //drawing a rectangle around the person
            cv::rectangle(frame, cv::Rect(cv::Point(x1, y1), cv::Point(x2, y2)),
                          cv::Scalar(0, 255, 0), 2);

            // Gets the coordinates in millimeters. Output these values to pi
            float x = det.spatialCoordinates.x;
            float y = det.spatialCoordinates.y;
            float z = det.spatialCoordinates.z;

            char label[64];
            snprintf(label, sizeof(label), "%.2f,%.2f\n", x, z);//print x value and z value

            //Send the x and z information to pi so that we can determine the rotation speed needed
for launch
            // and also know if we have centered onto the target.
        }
    }
        if(cv::waitKey(1) == 27) break;  // If the user presses ESC we exit
    }

    return 0;
}
```