```
// Read vision sensor to get angle needed to turn
// Returns angle to desired target
// Or 0 if error
double getRelativeAngle(int location = CENTER, int target = DEFAULT) {

    int lookingFor = BLUE_FLAG;     // default to red-team
    if (autonSelect == BLUEAUTON || autonSelect == BLUEBACKAUTON)
        lookingFor = RED_FLAG;      // but change to blue if needed

    if (target != DEFAULT)
        lookingFor = target;

    // Containers for the things we'll see
    std::vector<vision_object_s_t> blueThings;
    std::vector<vision_object_s_t> redThings;

    // Find number of objects visable
    int noObjs = camera.get_object_count();


    if (noObjs > 100)      // Camera error, so don't aim
        return 0;

    // Got through all objects seen
    for (int i = 0; i < noObjs; i++) {
        vision_object_s_t thisThing = camera.get_by_size(i);
        // Print their info

        // If object is a colour code
        if (thisThing.type == 1) {
            // Red flags should have angle ~0°
            if (thisThing.signature == RED_CODE_ID &&
             abs(thisThing.angle) < 90) {
                redThings.push_back(thisThing);
            }
            // Blue flags should have angle ~180°
            if (thisThing.signature == BLUE_CODE_ID &&
             abs(thisThing.angle) > 90) {
                blueThings.push_back(thisThing);
            }
        }
    }

    std::vector<vision_object_s_t> *theseThings;

    if (lookingFor == BLUE_FLAG)
        theseThings = &blueThings;
    if (lookingFor == RED_FLAG)
        theseThings = &redThings;

    if (theseThings->size() == 0)
        return 0;

    // Find which object is closest to left/middle/right
    double closestDist;
    if (location == CENTER) {
        closestDist = 10000;
        for (int i = 0; i < (*theseThings).size(); i++) {
            if (abs((*theseThings)[i].x_middle_coord -
             (VISION_FOV_WIDTH/2)) < closestDist) {
                closestDist = (*theseThings)[i].x_middle_coord;
            }
        }
    }
    if (location == LEFT) {
        closestDist = 10000;
        for (int i = 0; i < (*theseThings).size(); i++) {
            if ((*theseThings)[i].x_middle_coord < closestDist) {
                closestDist = (*theseThings)[i].x_middle_coord;
            }
        }
    }
    if (location == RIGHT) {
        closestDist = -10000;
        for (int i = 0; i < (*theseThings).size(); i++) {
            if ((*theseThings)[i].x_middle_coord > closestDist) {
                closestDist = (*theseThings)[i].x_middle_coord;
            }
        }
    }

    // Aim at the edge of the flag for better chance of toggleing
    if (lookingFor == RED_FLAG) closestDist += FLAG_OFFSET;
    if (lookingFor == BLUE_FLAG) closestDist -= FLAG_OFFSET;

    closestDist = closestDist - (VISION_FOV_WIDTH/2);

    if ((-closestDist/VISION_SEEK_RATE) == 0)
        return 0.001;

    return -closestDist/VISION_SEEK_RATE;

}
```