```cpp
// Read vision sensor to get angle needed to turn
double getRelativeAngle(int location = CENTER, int target = DEFAULT) {

    // Default to red-team
    int lookingFor = BLUE_FLAG;
    // But change to blue if needed
    if (autonSelect == 1)
        lookingFor = RED_FLAG;
    // Check if target isn't default
    if (target != DEFAULT)
        lookingFor = target;

    // Create vector of vision objects to store results
    std::vector<vision_object_s_t> allThings;

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

    // If there are more objects than possible flags, then error &
     return 0
    if (noObjs > 27)
        return 0;

    // Go through all objects
    for (int i = 0; i < noObjs; i++) {
        // Get the object
        vision_object_s_t thisThing = camera.get_by_size(i);
        // Check if the object from the camera matches what we want to
         find
        if (thisThing.signature == lookingFor)
            // If yes, add to vector
            allThings.push_back(thisThing);
    }

    // No correct objects found, so don't aim
    if (allThings.size() == 0) return 0;

    // Now check objects to delete any imposters - or don't aim if too
     close
    for (int i = 0; i < allThings.size(); i++) {
        if (lookingFor == BLUE_FLAG || lookingFor == RED_FLAG) {
            // Check if too big/close
            if (allThings[i].width > MAX_FLAG_WIDTH ||
             allThings[i].height > MAX_FLAG_HEIGHT) {
                allThings.erase(allThings.begin() + i);
            }
            // Check if too low
            if (allThings[i].y_middle_coord < MIN_FLAG_Y) {
                allThings.erase(allThings.begin() + i);
            }
        }
    }

    // Check if we've deleted all the things
    if (allThings.size() == 0) return 0;

    // Now find the thing furthest right or left, or the closest to
     the center
    double closestDist;
    if (location == CENTER) {
        // Start with some large distance
        closestDist = 10000;
        // Check all objects to check if closer
        for (int i = 0; i < allThings.size(); i++) {
            if (abs(allThings[i].x_middle_coord) < closestDist) {
                // Remember the closest dist
                closestDist = allThings[i].x_middle_coord;
            }
        }
    }
    if (location == LEFT) {
        // Start with some large distance
        closestDist = 10000;
        // Check all objects to check if further left
        for (int i = 0; i < allThings.size(); i++) {
            if (allThings[i].x_middle_coord < closestDist) {
                // Remember the furthest left
                closestDist = allThings[i].x_middle_coord;
            }
        }
    }
    if (location == RIGHT) {
        // Start with some large negative distance
        closestDist = -10000;
        // Check all objects to check if further left
        for (int i = 0; i < allThings.size(); i++) {
            if (allThings[i].x_middle_coord > closestDist) {
                // Remember the furthest right
                closestDist = allThings[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;

    // Scale by seek rate, and return the negative of the angle
    // Since positive rotation is CCW
    return -closestDist/VISION_SEEK_RATE;
}
```