```
// Read vision sensor to get angle needed to turn
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;

    std::vector<vision_object_s_t> allThings;
    std::vector<vision_object_s_t> greenThings;

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

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

    for (int i = 0; i < noObjs; i++) {                          // Go
     through them all
        vision_object_s_t thisThing = camera.get_by_size(i);    // And
         check if we care about
        if (thisThing.signature == lookingFor) {                // The
         type of object it is
            allThings.push_back(thisThing);                     // And
             stick it into a vector
        }
        if (thisThing.signature == GREEN_FLAG) {                // Put
         any green objects into vector
            greenThings.push_back(thisThing);
        }
    }


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

    // Check for any very large objects
    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);
                i--;
            }
        }
    }
```

```
#ifdef USE_GREEN_FLAGS
    // Now check objects to delete any imposters
    for (int i = 0; i < allThings.size(); i++) {
        if (lookingFor == BLUE_FLAG || lookingFor == RED_FLAG) {
            // Check if green object is within range
            bool greenWithinRange = false;
            for (int j = 0; j < greenThings.size(); j++) {
                // Reference frame is 0,0 top left, increasing right
                 and down D:
                if (lookingFor == BLUE_FLAG) {
                    if (
                        greenThings[j].y_middle_coord >
                         allThings[i].top_coord
                        &&
                        greenThings[j].y_middle_coord <
                         allThings[i].top_coord + allThings[i].height
                        &&
                        greenThings[j].left_coord +
                         greenThings[j].width <
                         allThings[i].x_middle_coord
                        &&
                        greenThings[j].left_coord >
                         allThings[i].left_coord - allThings[i].width
                        ) {
                        greenWithinRange = true;
                        break;
                    }
                }
                else {
                    if (
                        greenThings[j].y_middle_coord >
                         allThings[i].top_coord
                        &&
                        greenThings[j].y_middle_coord <
                         allThings[i].top_coord + allThings[i].height
                        &&
                        greenThings[j].left_coord >
                         allThings[i].x_middle_coord
                        &&
                        greenThings[j].left_coord +
                         greenThings[j].width <
                         allThings[i].left_coord + allThings[i].width
                         + allThings[i].width
                        ) {
                        greenWithinRange = true;
                        break;
                    }
                }
            }

            if (!greenWithinRange) {// && greenThings.size() > 0) {
```

```
                allThings.erase(allThings.begin() + i);
                i--;
            }

        }
    }
#endif

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

    // Now find the thing furthest right or left, or the closest to
     the center

    double closestDist;
    if (location == CENTER) {
        closestDist = 10000;
        for (int i = 0; i < allThings.size(); i++) {
            if (abs(allThings[i].x_middle_coord - (VISION_FOV_WIDTH/
             2)) < closestDist) {
                closestDist = allThings[i].x_middle_coord;
            }
        }
    }
    if (location == LEFT) {
        closestDist = 10000;
        for (int i = 0; i < allThings.size(); i++) {
            if (allThings[i].x_middle_coord < closestDist) {
                closestDist = allThings[i].x_middle_coord;
            }
        }
    }
    if (location == RIGHT) {
        closestDist = -10000;
        for (int i = 0; i < allThings.size(); i++) {
            if (allThings[i].x_middle_coord > closestDist) {
                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;

    closestDist = closestDist - (VISION_FOV_WIDTH/2);

    return -closestDist/VISION_SEEK_RATE;

}
```