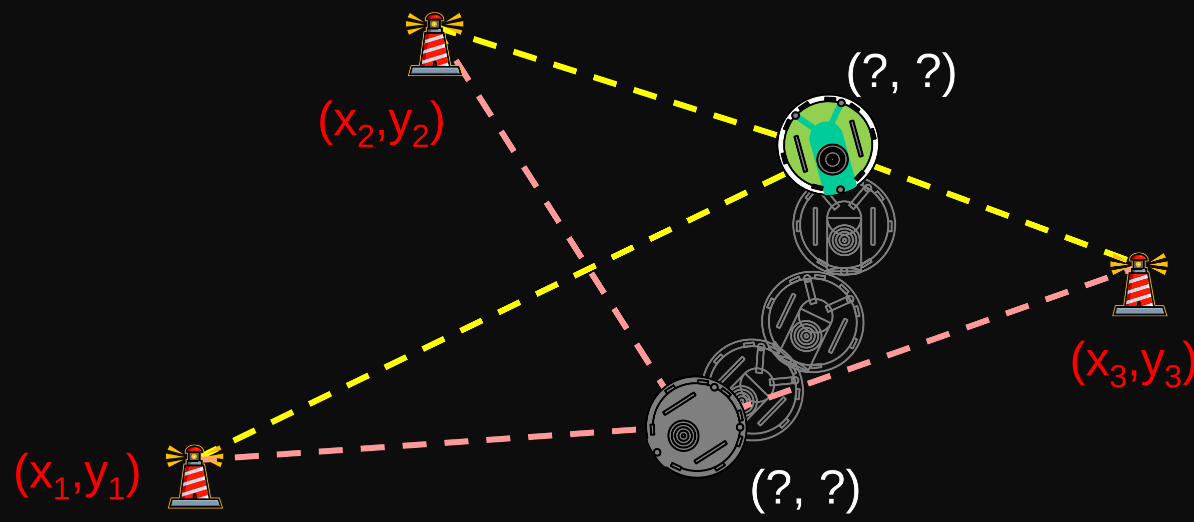
Beacon Positioning

(Triangulation)

Beacon Positioning

- A ***beacon*** is a detectable device that is placed at a fixed (i.e., known) position in the environment.
  - An ***active beacon*** transmits and/or receives signals

- A robot can estimate its “absolute” position and orientation by determining the distance and/or angle to three or more beacons.



Beacon Systems

- Active Beacon systems …

+ can produce high accuracy in position estimation

- can be expensive to install and maintain
- require alterations to environment (i.e., installation)

Image from http://cricket.csail.mit.edu/

- There are many commercially available indoor beacon systems:
  - Active Badge
  - Active Bat
  - RADAR
  - RICE project
  - E911
  - Cricket
  - MotionStar Magnetic Tracker
  - Easy Living
  - Smart Floor

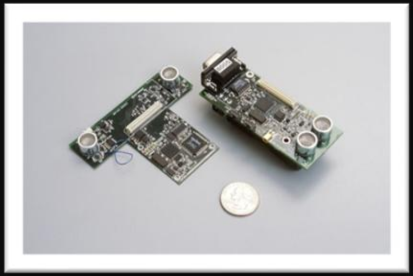
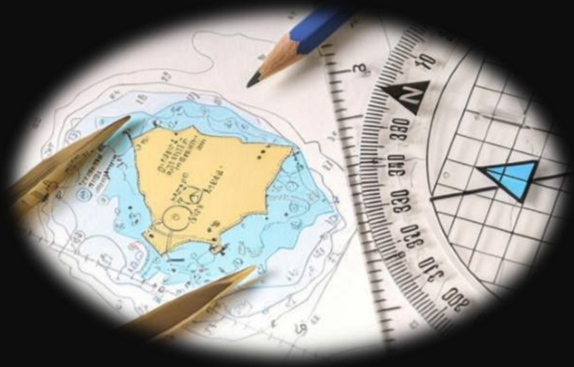


Image from http://cricket.csail.mit.edu/

Triangulating a Robot's Position

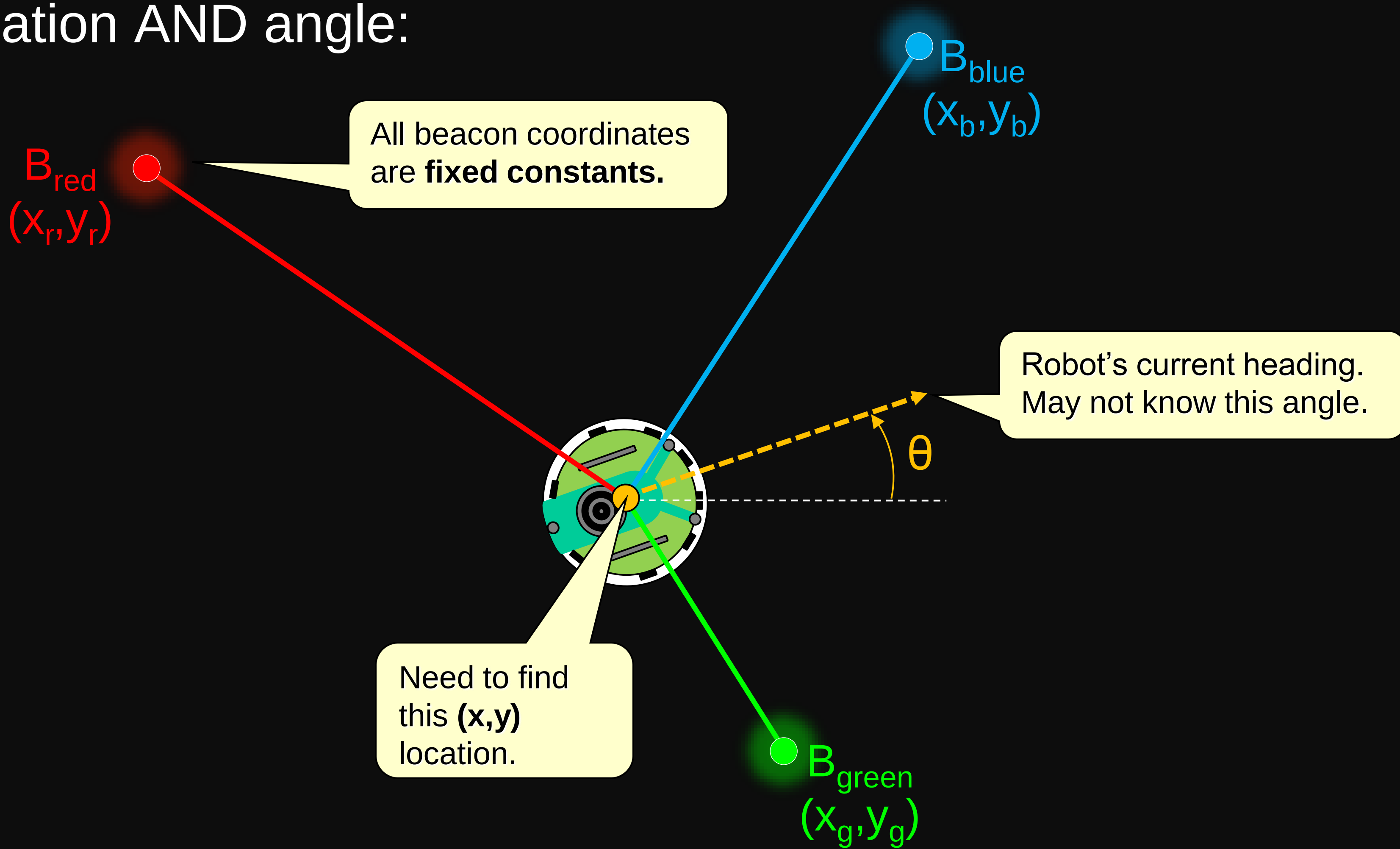
- Based on triangulation principle
  - uses geometric properties of triangles to compute position.



- Two types of triangulation techniques:
  - **Tri-Angulation**
    - determine robot <u>position</u> and <u>angle</u> based on angle to beacons
    - 2D requires 2 angles and 1 known distance, or 3 angles.
    - 3D requires 1 additional azimuth measurement
  - **Tri-Lateration**
    - determine robot <u>position</u> based on distance from beacons
    - 2D requires 3 non-colinear points
    - 3D requires 4 non-coplanar points

Triangulation

- <u>Problem:</u> Given 3 beacons at fixed positions, compute robot location AND angle:



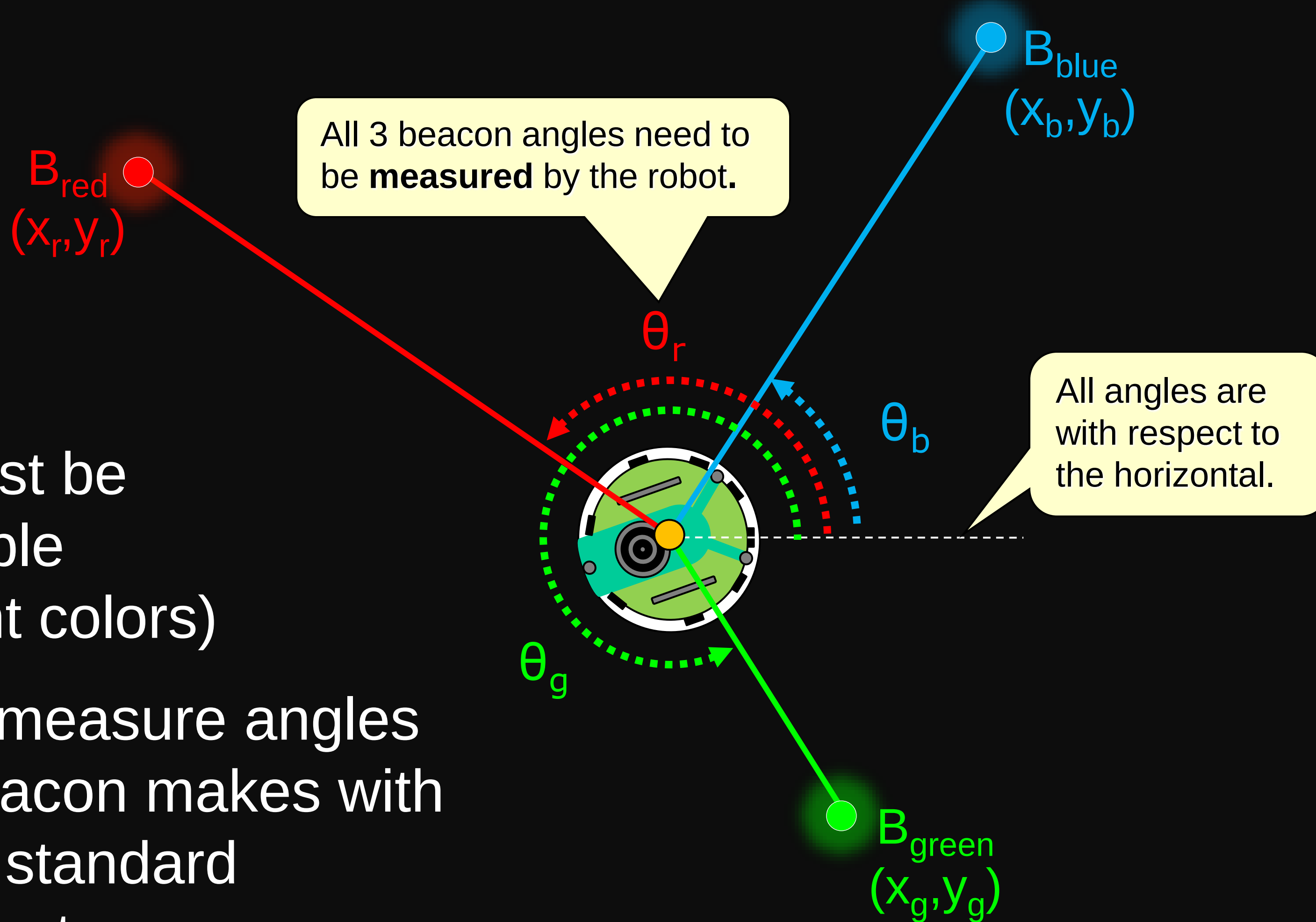
Triangulation - Techniques

- There are many triangulation techniques for mobile robot positioning.
  - some require a particular beacon ordering
  - some have “blind spots”
  - some only work within the triangle defined by the three beacons
  - more reliable methods exist but are more complex or require the handling of certain spatial arrangements separately.

- We will use the “To Tal” algorithm:
  - V. PIERLOT, M. VANDROOGENBROECK, and M. Urbin-Choffray. **A new three object triangulation algorithm based on the power center of three circles**. In Research and Education in Robotics (EUROBOT), volume 161 of Communications in Computer and Information Science, pages 248-262, 2011. Springer.

Triangulation – The *ToTal* Algorithm

- Assumes 3 fixed beacon locations:



- beacons must be distinguishable (e.g. different colors)
- Robot must measure angles that each beacon makes with reference to standard coordinate system

Triangulation – The *ToTal* Algorithm

- Don't worry about the math. Just need to plug in formulas:
  - STEP 1: compute the modified beacon coordinates …

$$x'_1 = x_r - x_g$$
$$y'_1 = y_r - y_g$$
$$x'_3 = x_b - x_g$$
$$y'_3 = y_b - y_g$$

  - STEP 2: compute the three cotangents …

$$T_{12} = 1 / \tan(\theta_g - \theta_r)$$
$$T_{23} = 1 / \tan(\theta_b - \theta_g)$$
$$T_{31} = \frac{1 - T_{12} * T_{23}}{T_{12} + T_{23}}$$

Triangulation – The *ToTal* Algorithm

- STEP 3: compute the modified circle center coordinates…

$$x'_{12} = x'_1 + T_{12} * y'_1$$
$$y'_{12} = y'_1 - T_{12} * x'_1$$
$$x'_{23} = x'_3 - T_{23} * y'_3$$
$$y'_{23} = y'_3 + T_{23} * x'_3$$
$$x'_{31} = (x'_3 + x'_1) + T_{31} * (y'_3 - y'_1)$$
$$y'_{31} = (y'_3 + y'_1) - T_{31} * (x'_3 - x'_1)$$

- STEP 4: compute k'_{31}…

$$k'_{31} = x'_1 * x'_3 + y'_1 * y'_3 + T_{31} * (x'_1 * y'_3 - x'_3 * y'_1)$$

- STEP 5: compute d …

$$d = (x'_{12} - x'_{23}) * (y'_{23} - y'_{31}) - (y'_{12} - y'_{23}) * (x'_{23} - x'_{31})$$

Triangulation – The *ToTal* Algorithm

– if d = 0, then there is no solution

- corresponds to situation when robot is on perimeter of circle defined by beacons
- no algorithm can handle this case
- can detect when this happens by examining d.
- As d gets smaller, error grows … so if, for example, | d | < 100 or so … then the calculation may be inaccurate.

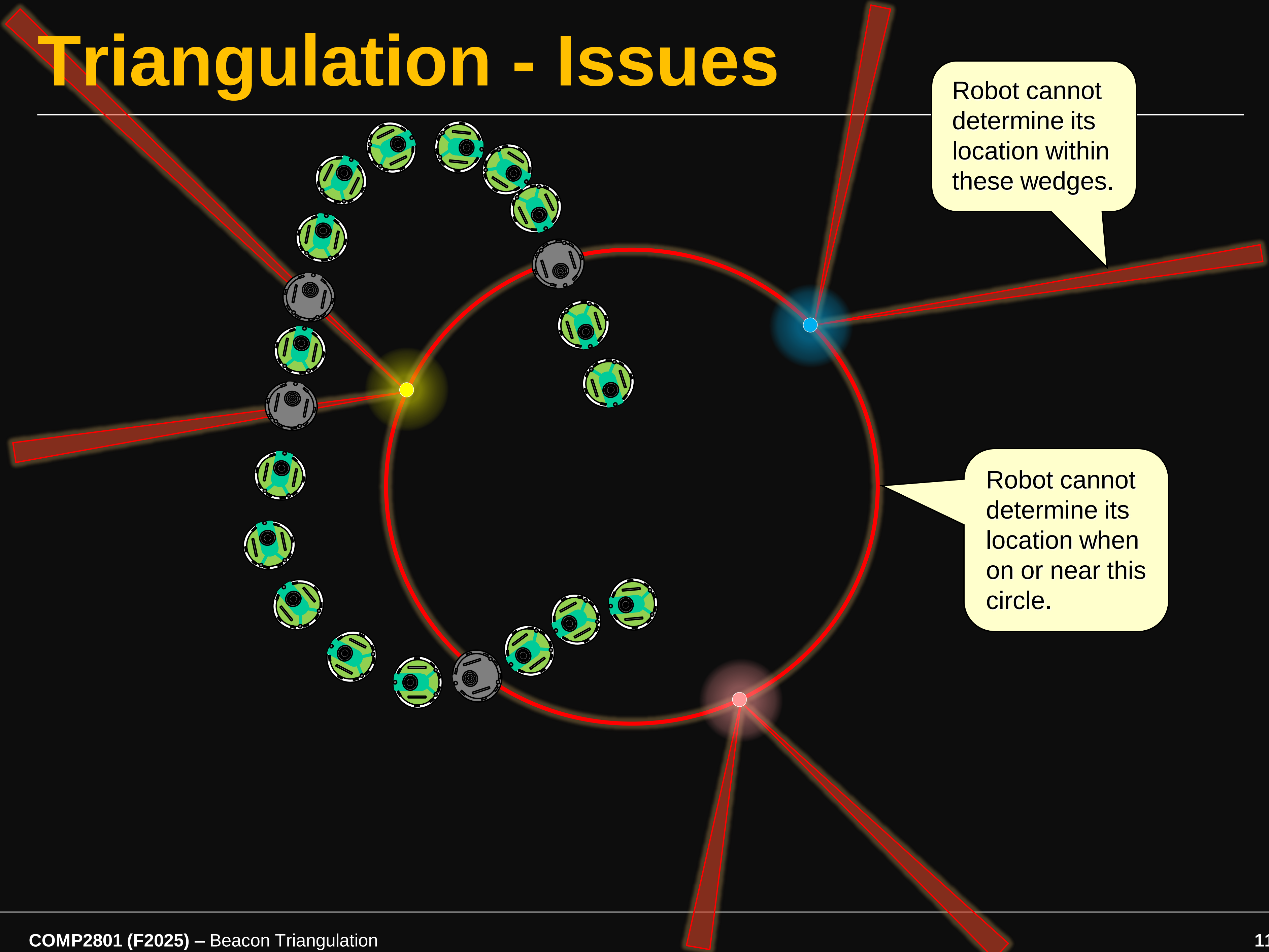
```
if (Math.abs(d) < 100)
     ...
```

– STEP 6: compute the robot position (x, y) …

$$x = x_g + k'_{31} * (y'_{12} - y'_{23}) / d$$

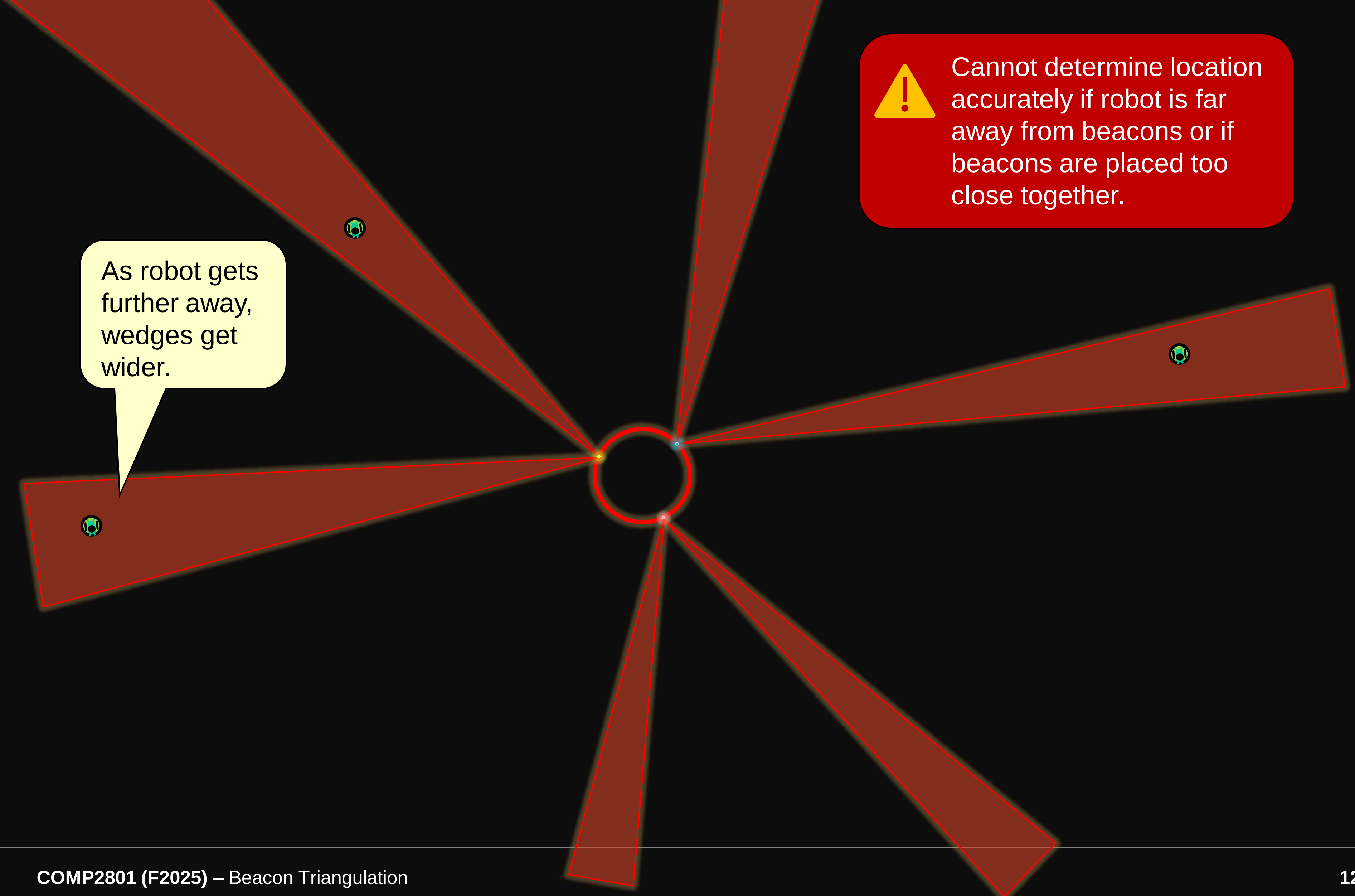
$$y = y_g + k'_{31} * (x'_{23} - x'_{12}) / d$$

Triangulation - Issues



Triangulation - Issues

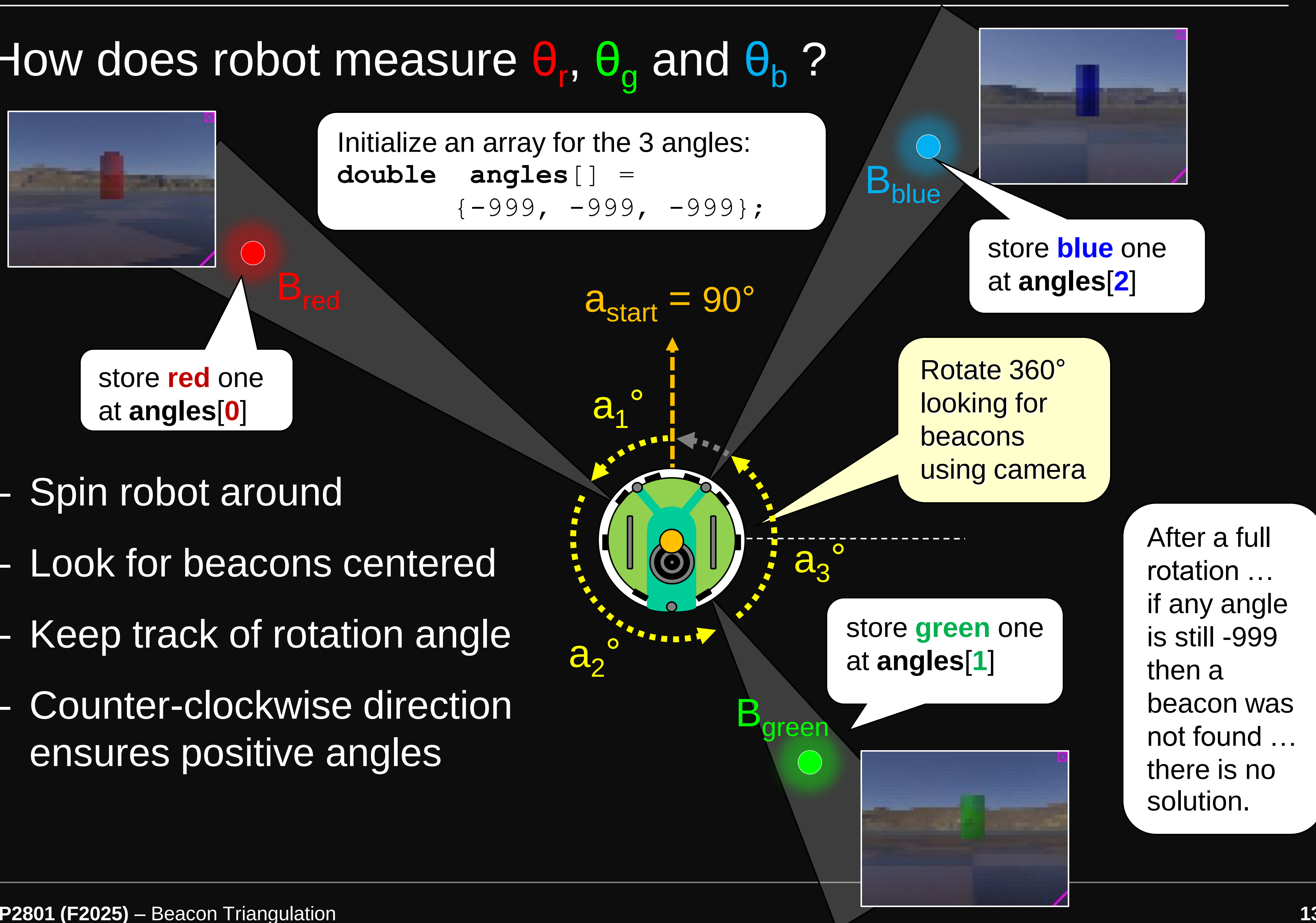
Cannot determine location accurately if robot is far away from beacons or if beacons are placed too close together.



As robot gets further away, wedges get wider.

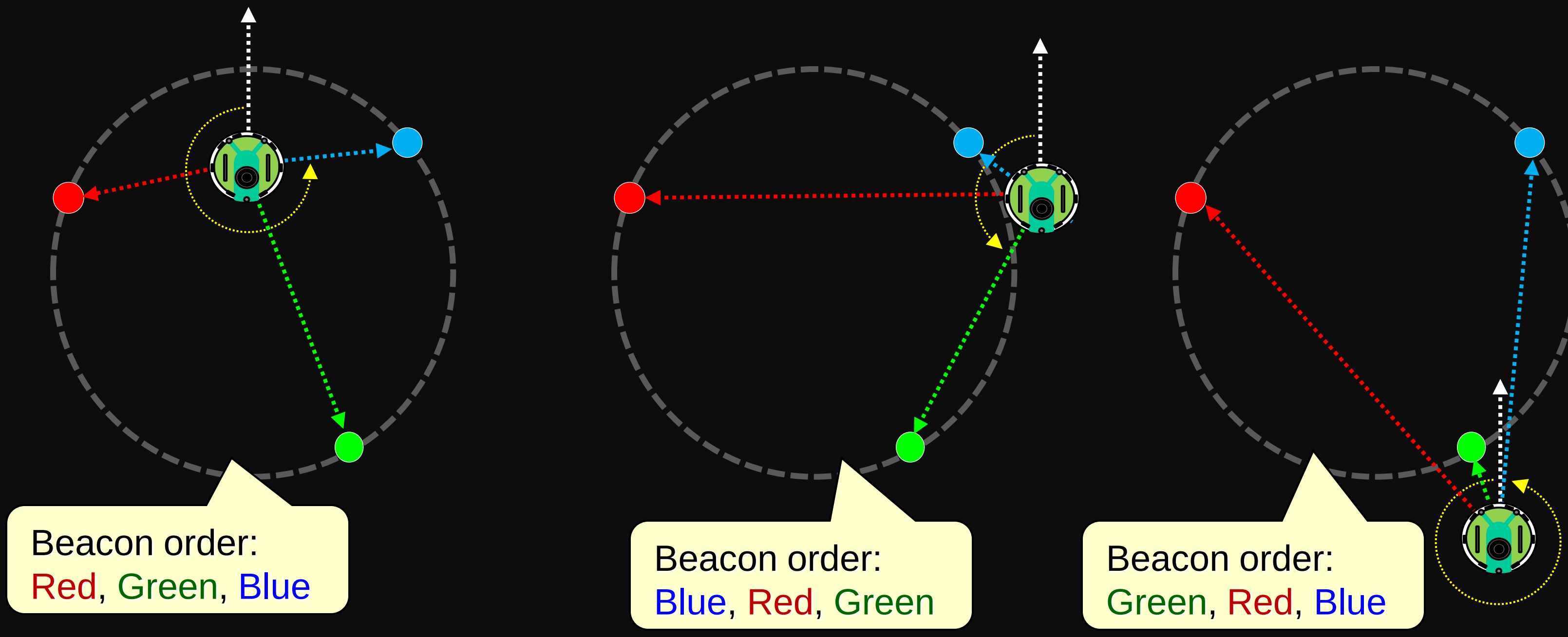
E-Puck - Measuring Angles

- How does robot measure θ_r, θ_g and θ_b ?
  - Spin robot around
  - Look for beacons centered
  - Keep track of rotation angle
  - Counter-clockwise direction ensures positive angles



E-Puck – Beacon Ordering

- Keep in mind, the beacons may be encountered in a different order each time, depending on robot's location:



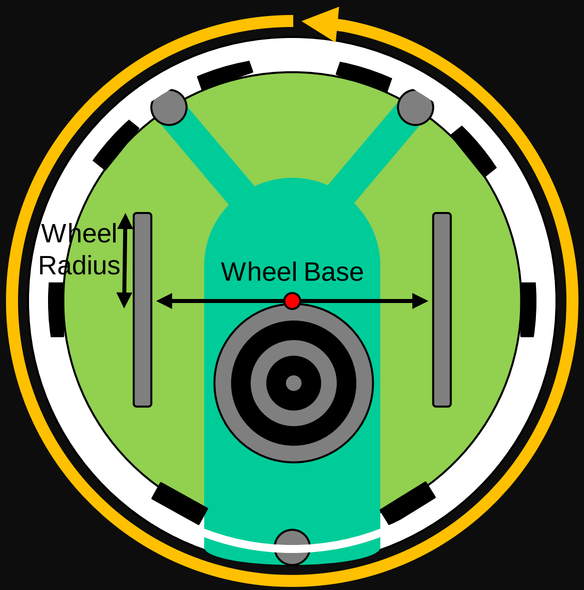
- In the array, just make sure that red goes at position 0, green at position 1 and blue at position 2.

E-Puck - Measuring Angles

- Use wheel position sensors to measure angle while spinning
- Need to rotate 360°
- While rotating, look for the beacons

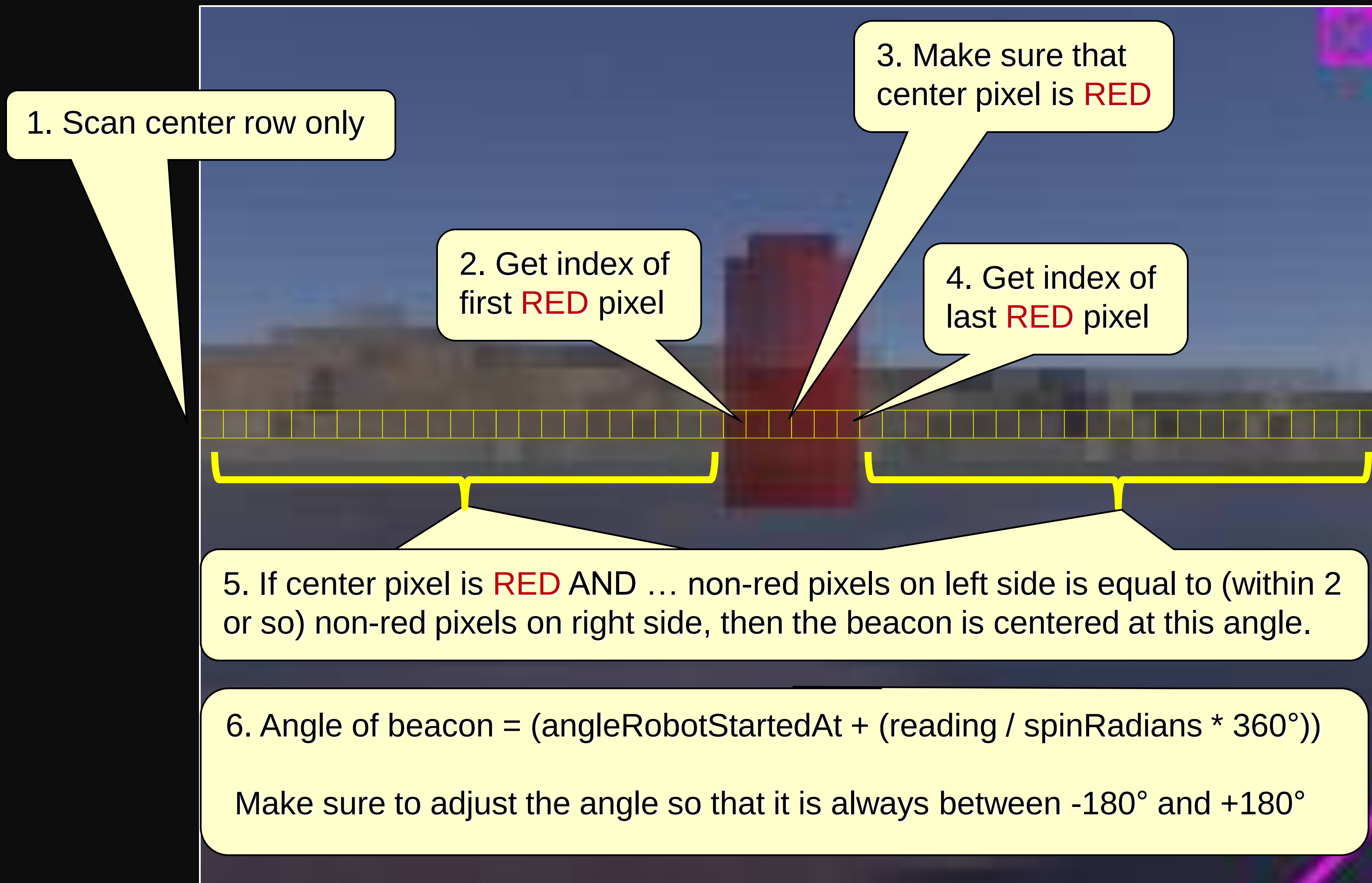
spinRadians is the number of radians that the wheels should turn in order for the robot to spin a full circle.

```
spinRadians = PI * WHEEL_BASE / WHEEL_RADIUS
Start rotating robot counter-clockwise
reading = 0;

while (reading < spinRadians) {
   reading = readWheelSensor - previousReading;

   // CODE TO LOOK FOR BEACON GOES HERE
   // STORE THE THREE ANGLES  θr, θg and θb

}

Stop rotating robot
previousReading = readWheelSensor;

Compute the position (x,y) using triangulation method
```



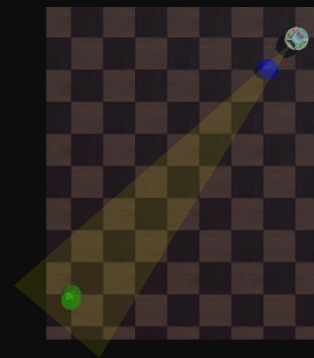
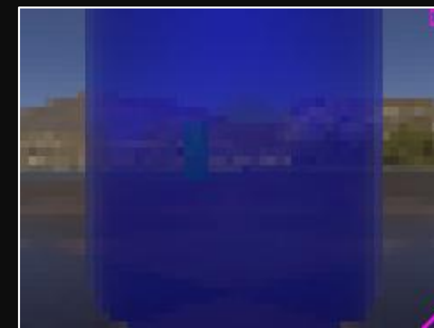
E-Puck - Measuring Angles

- How to check if a beacon is centered at this angle:



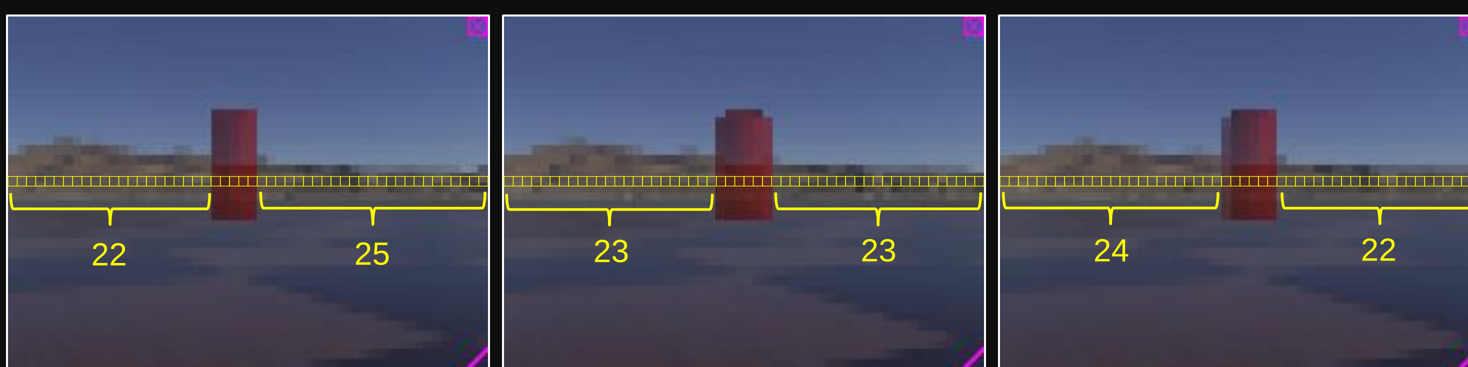
E-Puck - Measuring Angles

- You will need to check individually for the green and blue beacons in the same manner.

- It is possible that robot cannot see one of the beacons.
- It is also possible that the robot may see a beacon two or three times … so just remember the first time it sees it centered (e.g., use a **boolean** flag foundRed)

When robot spins CCW, beacon appears to move right



Start the Lab …