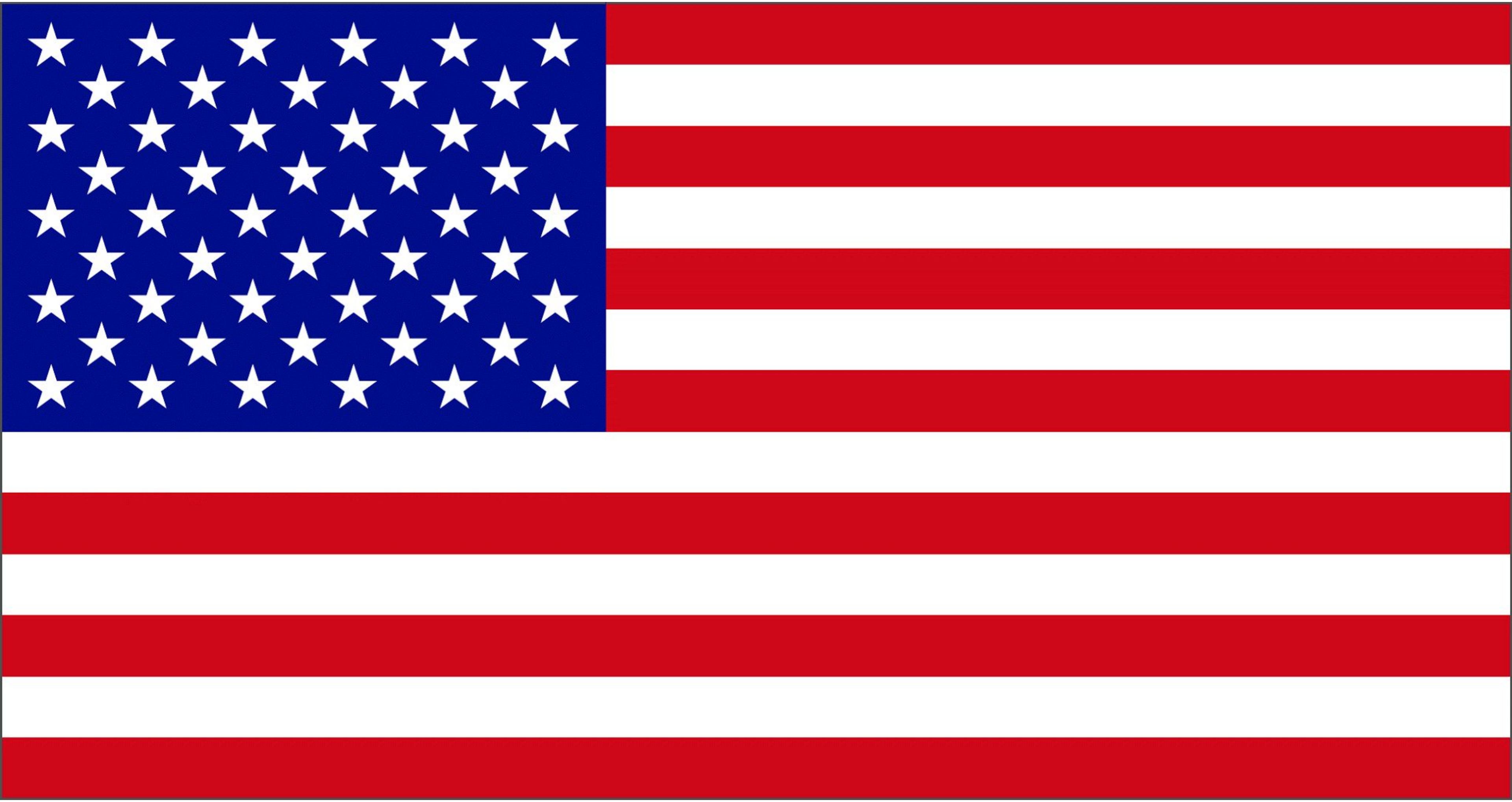
2016-2017 Season

www.ashlandroboticsclub.com

RADICAL

ROBOTS



H
H
H
H
18 U.S.C.707

Agenda

- Fundraiser
- Website updated
- Discuss books and posters
- Group project judging will take place after competition (at pizza party?)
    - Going to try to bring in impartial judge(s)
- Upcoming officer elections (prepare your speeches!)
    - Introduce yourself to the club and tell why you would be a good officer
- Strategy discussions
- Discussion of designs and pseudo code
    - What is and isn’t accpetable
- Time to work on your plans (robot designs and pseudo code.)

Website Updated

www.ashlandroboticsclub.com

- Will serve as an archive and document repository
- Facebook will still be used for notifications (along with newsletter reminders of meetings)
- Meeting slideshow presentations will be uploaded after the meeting for access later

Books and Posters

- Club will provide books
- Posters should be ½ size of a poster board
- Books and posters must be completed before competition registration in February
    - We will **NOT** be working on these during meetings
- Once books are completed, they should be kept in your folder for judging
- Judging will take place after the competition (at pizza party?)

Officer Elections

- Anyone who is a 2nd year member or more is eligible
- Must give a brief (at least 1 minute) speech to introduce themselves to the club (we have a lot of new members) and tell why they would be a good officer
    - Try to avoid giving a speech for a specific office!
- You may decline to run for specific offices
- We will be electing the following officers:
    - President
    - Vice President
    - Secretary
    - Treasurer
    - Reporter

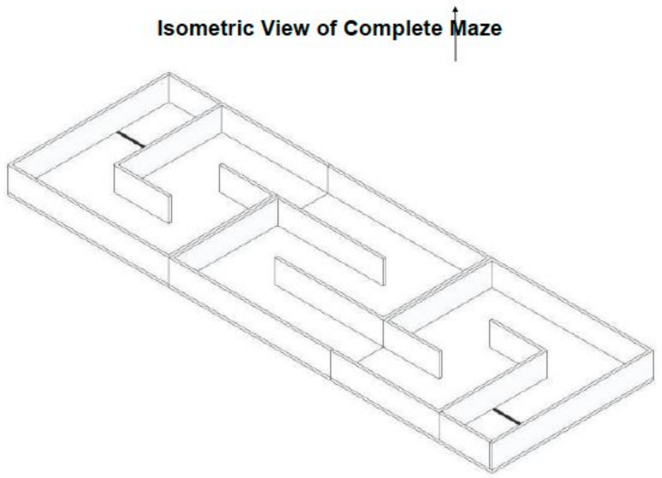
Strategy

Strategy

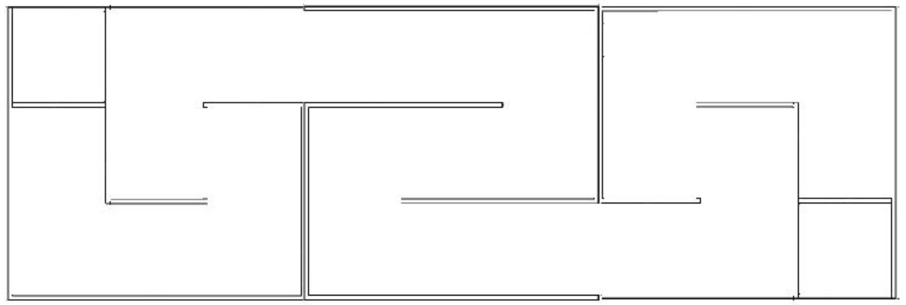
Maze

Elementary / Middle School

Isometric View of Complete Maze



Top View of Complete Maze



Typical Strategy

- Move straight until the robot senses (ultrasonic or touch) a wall, then turn 90 degrees
- Repeat for each section of maze until finished

PROBLEMS:

- Robots do not always move perfectly straight
  - Wheels can be slightly off, causing them to veer to the side
  - Motors can be slightly different, causing wheels to move at different speeds
- Difficult to turn perfectly 90 degrees
  - Distance between wheels affects how much they need to rotate in order to turn (circumference = pi * diameter)
- If robot isn't aimed perfectly, they can (eventually) run into a wall)

Alternative Strategy

- Addition of a sensor on the side of the robot
  - Example: Use an ultrasonic sensor to detect how far from the wall the robot is. Use this to keep them within a safe distance, and use a front sensor to detect when to turn.
  - Will require additional loops or conditionals within the code
  - Turns can be approximately 90 degrees, and the extra sensor will take care of making sure the robot doesn’t hit the wall.

Strategy

Sumo

Typical Strategy

- Turn left or right until the robot “sees” the opponent, then move straight until it detects the white line, back up and start over.
- Try to make the robot as heavy as possible

PROBLEMS:

- May not always see your opponent (e.g. smaller bots)
- If you need to turn 270 degrees, and you opponent only needs to turn 90 degrees, they will see you first (giving them an advantage)

Alternate Strategies

- Use multiple programs to turn the shortest distance possible
  - Example: If your opponent is to your left, turn left. If they are on your right, turn right. If they are behind you, you could devise a program that just initially backs up! (Would need to be careful not to go out-of-bounds.)
- Use evasion techniques!
  - Rather than trying to push your opponent right away, move forward (or backward) right away. If they are expecting you in a specific location, you'll throw them off!

Alternate Strategies

- Move fast!!!
    - When searching for your opponent, move quickly to find them first
    - When pushing, move as quick as you can (so they don’t escape)
- Don’t rely on the sensors
    - The ultrasonic sensor might not always find your opponent
    - Might be good to have a backup plan where if the robot turns at least 1 full turn, it just moves straight until it sees the line, then start over.
- Be careful with the boundaries!!!
    - If any part of your robot goes out of bounds, you’ll lose. Make sure the sensor is detecting the line before the corners go past the line!

Advanced Strategies

- Try to fool the ultrasonic sensors
  - Indented foam can fool weaker bots with higher thresholds for their ultrasonic sensors (http://www.botsfromscratch.com/mini-sumo)
- Use white tape on your ramps to try to trick light sensors
- Try to get stickier wheels (difficult with LEGO kits, but possible)
- Try to get stronger, faster motors (again, difficult with LEGO kits, but definitely possible for those with some advanced technical skills.)

Advanced Strategies

- Read the following sites:
  - http://www.botsfromscratch.com/mini-sumo
    - This guy has competed in and won at the NRC
  - http://www.instructables.com/id/Android-Controlled-Bluetooth-Sumobot-Ultimate-DIY-/
    - Arduino-based robot; very formidable!

Strategy

Bot Ball

New for 2017 is the addition of a bonus tube with bonus ping-pong balls. You receive double points for each bonus ball in your gutter (10 points, vs. 5 points for the regular balls.) However, you'll lose 30 points for the bonus tube being in your gutter, and lose 20 points for the bonus tube being in your end zone.

It may be strategically advantageous to try to get the tube into the opponent's gutter, causing them to lose 30 points. However, keep in mind that you are not competing directly against the other person on the game floor with you. You are competing against ALL competitors based on your score.

Robot Designs and Psuedo Code

- You must provide a robot design and pseudo code before you will be assigned a robot. Even if you are basically using a previous robot, you MUST submit a design sketch.
- Designs do not need to be overly detailed, but they should show what the robot is supposed to look like. Sensors and any other attachments should be labeled, with a brief description of what they do.
- Pseudo code shouldn't be actual code, but it needs to accurately describe the program. It should be possible to give your pseudo code to anyone, and they should be able to complete the program.
    - You should specify loops and conditionals
    - Actual sensor values aren't necessary, but there should be reasonable values in the design process.

NOT Acceptable (Designs)

- Designs that are quick sketches with little/no details
- Drawings that are too small to see what is happening with the robot (feel free to use a full sheet for each angle)

Acceptable (Designs)

- Show all sensors or attachments you will be using and where they will be mounted
- Show as much detail as possible in the robot (we don’t expect you show every LEGO piece, but show enough that someone else could build the robot.
- Follow-through -- Your final robot should be close to your design (this will be part of your grade at judging.)

NOT Acceptable (Pseudo Code)

SUMO

```
Turn until we see opponent, move forward until we see the line, back up and
start over.
```

MAZE

```
Move forward, turn left, move forward, turn right (... etc.)
```

Acceptable (Pseudo Code)

- Use comments to label sections (You can use C-style comments (//) or anything else to distinguish them from actual 'code'
- Pseudo-code should show the full program (i.e. for the maze, you can't just show one section, however, you might be able to loop sections.)
- Should be detailed enough that someone else could write the actual program, but not so detailed as to BE the actual program.

Acceptable (Pseudo Code)

SUMO (sample)

```
// Opponent on my left
// Loop indefinitely
Loop
     // Look for opponent
     While opponent is not visible and wheel rotation is less than 4
          Move infinite rotations toward the left (slider dragged all the way to the left)
     EndWhile
     // Move toward line
     While light sensor doesn’t see white
          Move straight forward infinite rotations
     EndWhile
     // Backup
     Move straight backward 1 rotation
EndLoop
```

Acceptable (Pseudo Code)

MAZE (sample)

```
// Section 1
While front distance is not close
    If right distance is too much
        Move forward with faster speed on left motor
    Else
        Move forward with faster speed on right motor
    EndIf
EndWhile

// Turn left approximately 90 degrees
Move 1 rotation counter-clockwise  // Drag slider to the left
```