iRobot Create Behaviors and GUI

CSSE 120—Rose Hulman Institute of Technology

Thought for the week

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

R. Fowler, *Refactoring: Improving the Design of Existing Code*

Robot Behaviors

- *Behaviors* achieve and/or maintain particular goals
  - ***Wall following*** behavior maintains the goal of following a wall
  - ***Homing*** behavior achieves the goal of getting a robot to the home location
- *Behaviors* are not instantaneous but take time to achieve
- *Behaviors* can take input from sensors as well as other behaviors
- *Behaviors* are more complex than actions such as stop, turn right, move in circle

Wall Following:
Feedback Control

- In this project, you will develop a ***wall following*** *behavior* for the Create robot
- The sensor feedback will include the wall sensor and wall signal
  - Wall sensor (0 or1) – Indicates presence of wall
  - Wall signal (0 – 4095) – Indicates distance to wall
- If the goal is for the robot to maintain a certain distance from the wall while following, then the wall signal can be used to calculate the ***distance error***
- You will use the amount of distance error to determine the angle or speed with which the robot must adjust to maintain the required distance , this is called **feedback control**
- The amount of adjustment in angle or speed is called the **gain**
- Adjusting the robot's gain based upon distance error is called ***proportional control***

Wall Following: Two programming algorithms

Algorithm 1:

robot will oscillate a great deal

robot rarely if ever reaches the desired distance before getting too close to or too far from the wall

```
If distance-to-wall is exactly as desired,
     then keep going.
If distance-to-wall is larger than desired,
     then turn by 45 degrees toward the wall,
     else turn by 45 degrees away from the wall.
```

Algorithm 2:

robot will switch and wiggle back and forth as it moves along but less than before

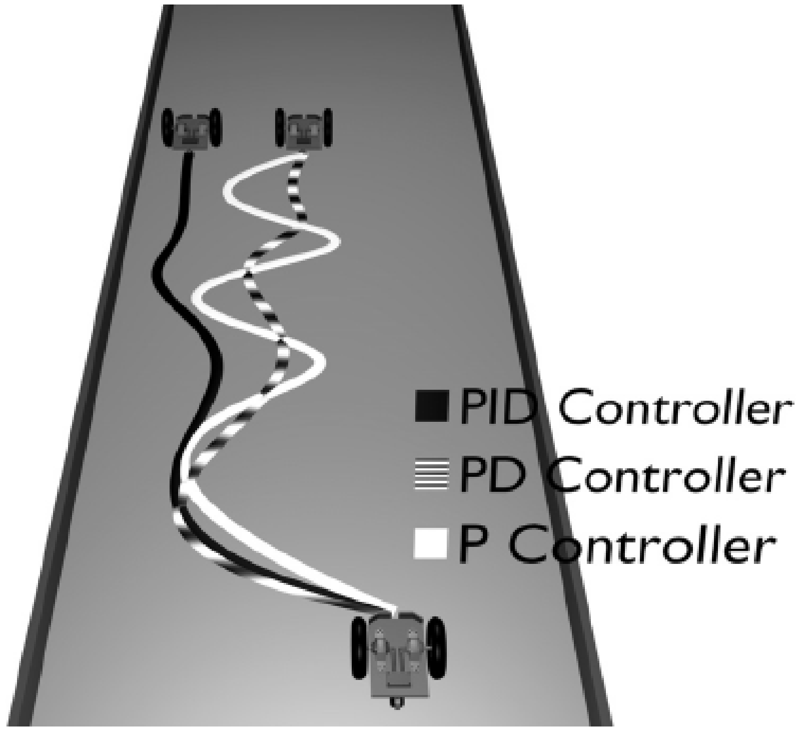
the amount of wiggling depends on how often the error is computed and how much correction is made

```
If distance-to-wall is in the right range,
     then keep going.
If distance-to-wall is larger than desired,
     then turn toward the wall,
     else turn away from the wall.
```

Wall Following (Feedback Control): Tuning parameters

To decrease oscillations:

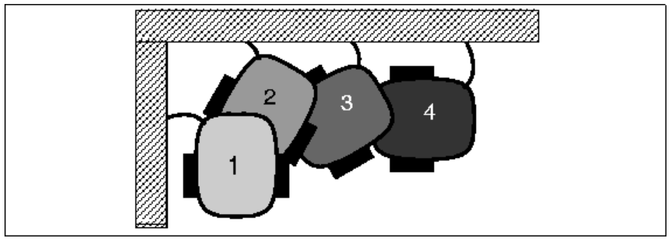
- Compute the error often so that the robot turns often
- Adjust the turning angle so that the robot turns by small rather than large angles
- Determining the proper amount to turn is called calibrating or **tuning the control parameters**
- Sometimes it takes trial and error to find the proper gain



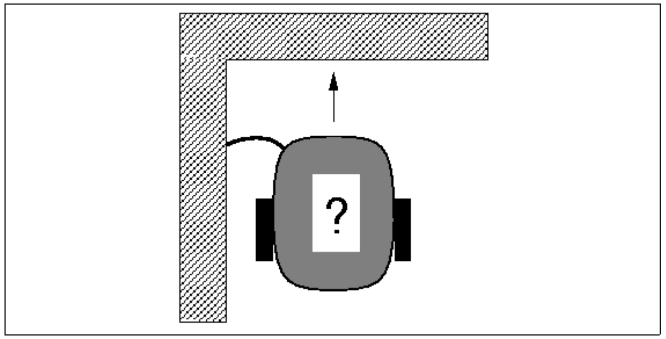
Wall Following:
Negotiating a corner

- Little Turns:
  - Make little turns, drive straight

  ahead, detect the wall, back up, repeat
  - The disadvantage is that this method is time consuming and produces jerky movements

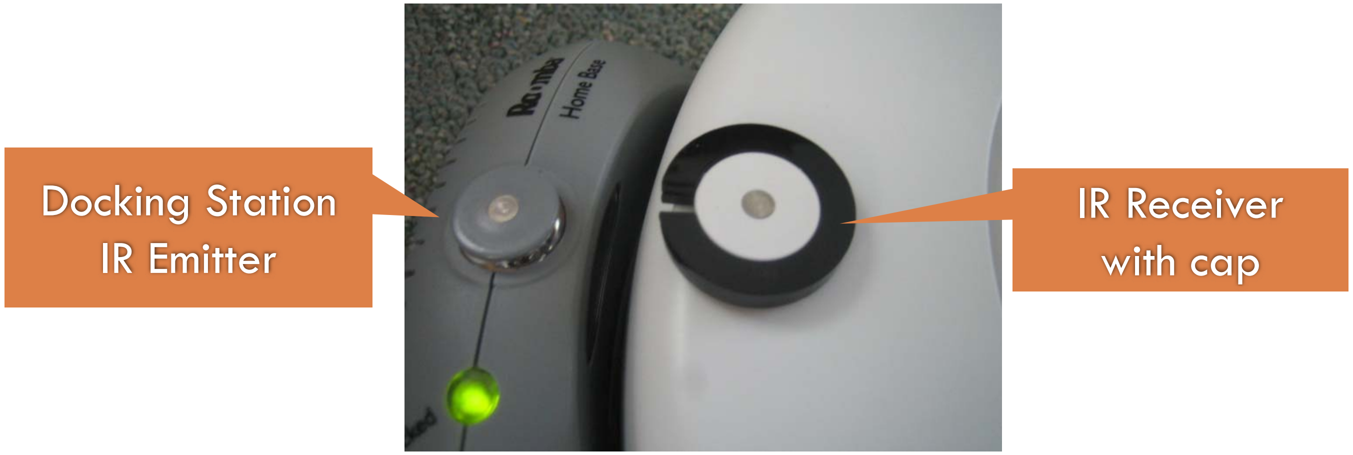


- 90° turn:
  - Execute a turn command that was timed to accomplish a ninety degree rotation (open loop control)
  - Works reliably only when the robot is very predictable (battery strength, traction on the surface, and friction)



Homing or Finding other Robots

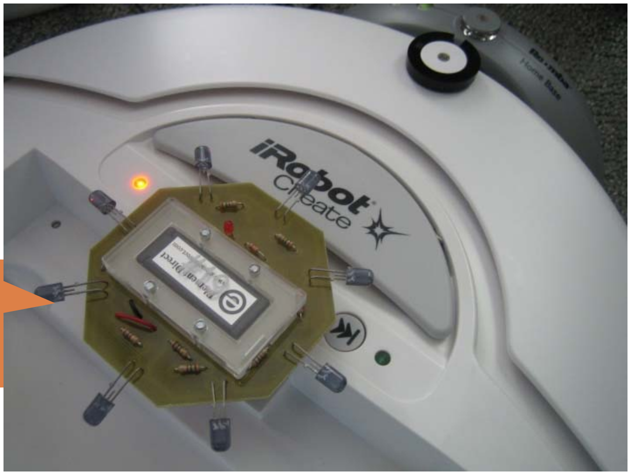
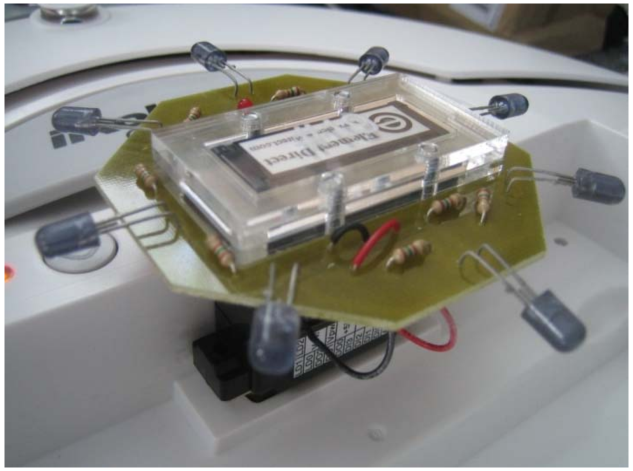
- Recall that the wall sensor is an infrared sensor to detect distance
- The wall sensor contains an emitter and receiver and uses the time of flight to calculate distance to the wall
- The Create robot also has an omnidirectional infrared receiver on top. This will be covered with a **cap** to become a unidirectional receiver to detect a robot signal in front



Finding other Robots:
IR Emitter

- Each robot will also have a BAM hat that emits an omnidirectional distinct signal via IR LED emitters
- This signal will enable other robots to find your robot
- The BAM hat will be tied to LD1 (black) and 5V (red) on the BAM to send out an IR Byte

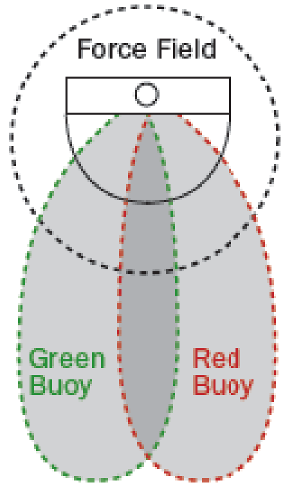
BAM hat:
IR Emitter

Finding other Robots:
IR signals

- Similar to the wall following, once the robot detects an IR signal, the control algorithm should determine the amount to turn and go forward
- If the signal is lost, the robot should pan to find the signal again
- The robot should iteratively repeat this process until it bumps into the other robot and stops
- Note that the other robot may also be moving
- To test your code use the IR bytes emitted from your docking station.
- The values are shown to the right for the different buoys
- Note that a value of **255** indicates no signal found

Dock beam configuration



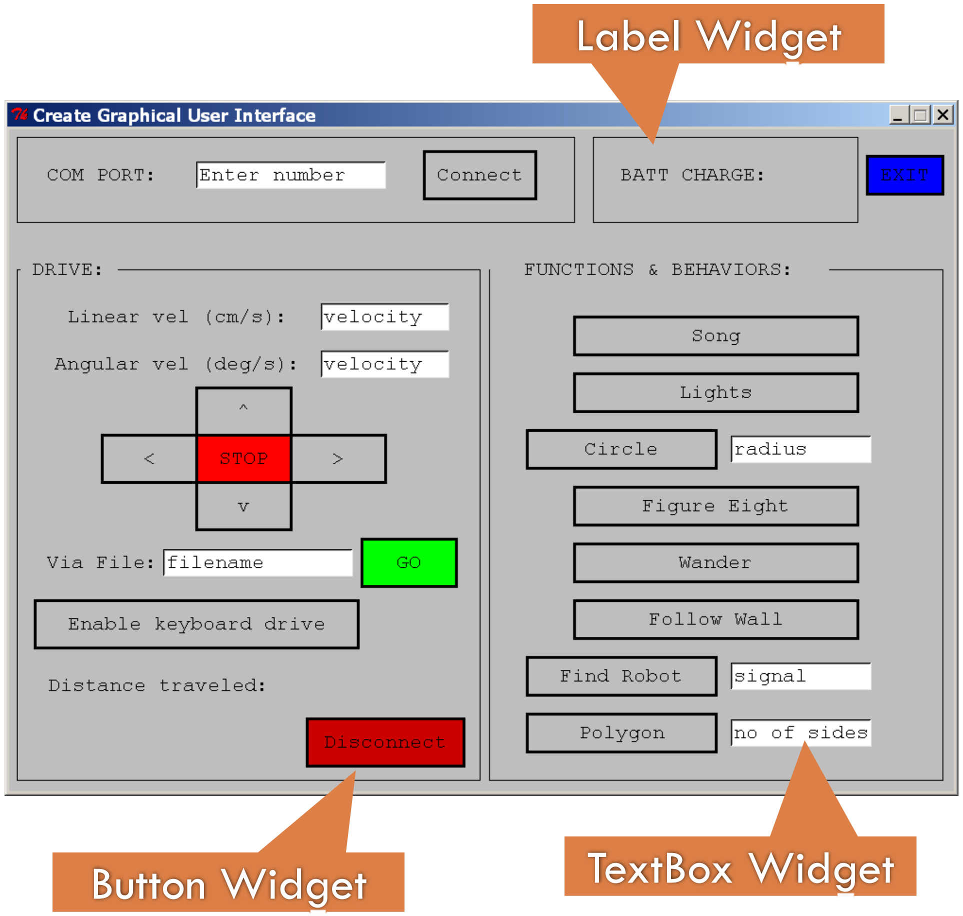
248	Red Buoy
244	Green Buoy
242	Force Field
252	Red Buoy and Green Buoy
250	Red Buoy and Force Field
246	Green Buoy and Force Field
254	Red Buoy, Green Buoy and Force Field

Graphical User Interface (GUI) Design

- The goal of user interface design is to create an interface with increased **usability.**
- **Usability** is the ease of use and efficiency of use of a graphical user interface
- The interface should be designed to be user-centered in order to afford and simplify task execution

Create Graphical User Interface

- **Label Widget**
  - Created from rectangle and text
  - Labels GUI widgets
  - Displays distance and battery charge
- **Button Widget**
  - Created from rectangle and text
  - Input to drive robot
  - Input to connect to robot
  - Input to execute behaviors and functions
- **TextBox Widget**
  - Created from entry
  - Enter robot velocities
  - Enter move to filename
  - Enter circle radius
  - Enter polygon number of sides
  - Enter robot signal


Label Widget
Create Graphical User Interface
COM PORT:
Enter number
Connect
BATT CHARGE:
EXIT
DRIVE:
Linear vel (cm/s):
velocity
Angular vel (deg/s):
velocity
^
<
STOP
>
v
Via File:
filename
GO
Enable keyboard drive
Distance traveled:
Disconnect
FUNCTIONS & BEHAVIORS:
Song
Lights
Circle
radius
Figure Eight
Wander
Follow Wall
Find Robot
signal
Polygon
no of sides
Button Widget
TextBox Widget