Search and Pursuit-Evasion

Team Swift & Stealth

Alex Melnick
Maura Mulligan
Megha Shah
Mitch Hornak

Project Motivation

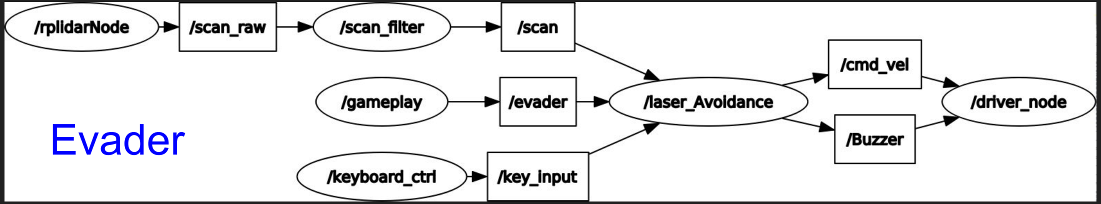
- **Search and Pursuit-Evasion** is a key challenge in robotics with critical applications in **Defense** and **Search-and-Rescue**.
- **Motivation**: Explore advanced robotic interactions in dynamic scenarios while gaining experience with industry-standard tools like **ROS**.
- **Goal**: Program an autonomous robot with **multisensor integration** to perform Search and Pursuit-Evasion of a human-controlled robot.

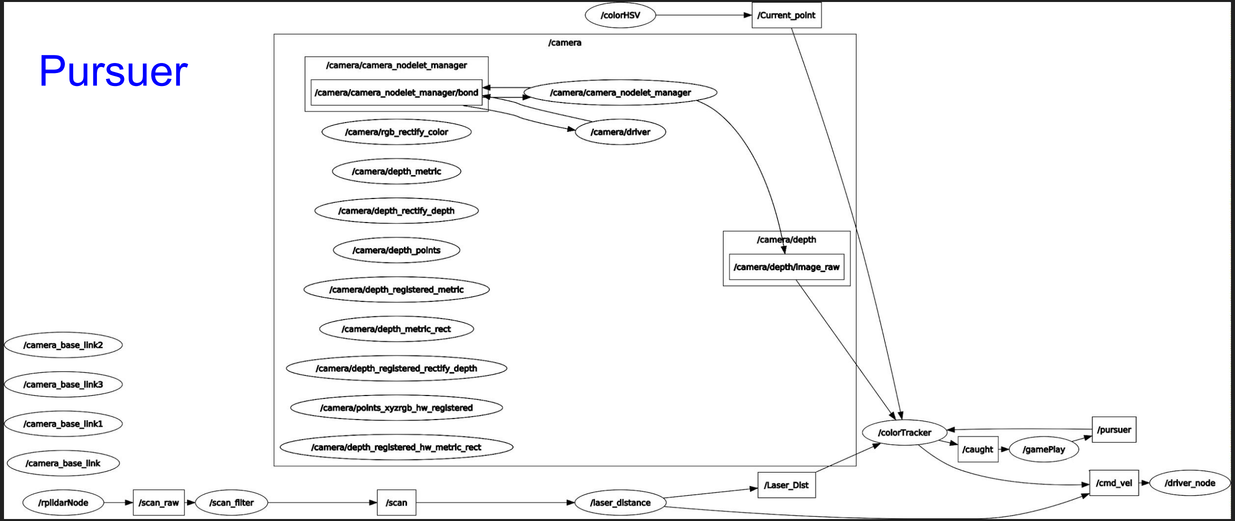
ROS

Design Diagram - ROS

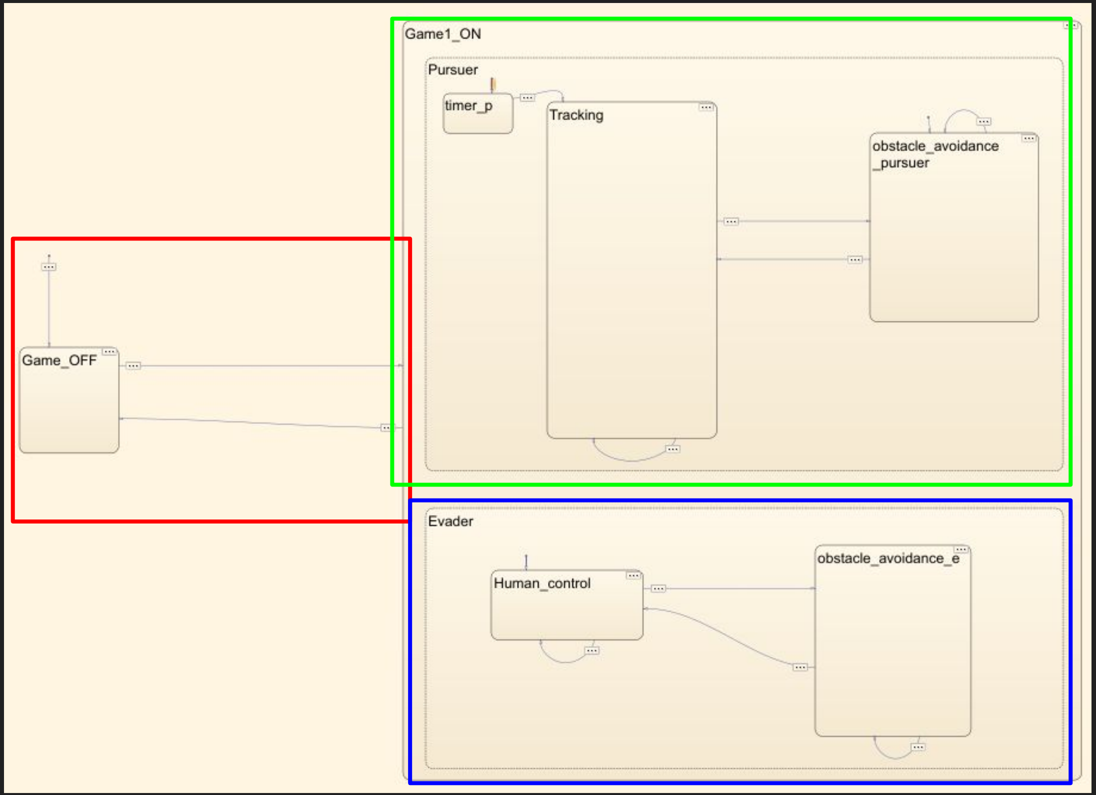
Hardware: Yahboom ROSMASTER X3 featuring LiDAR and color depth camera

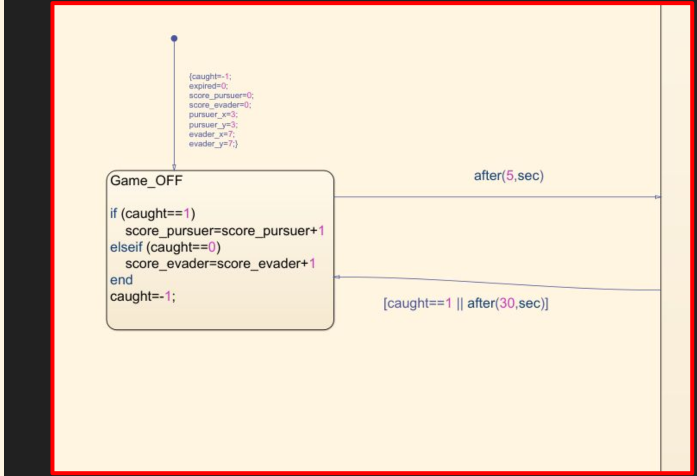
Software: ROS1 Melodic and Python





Design Diagram Overview

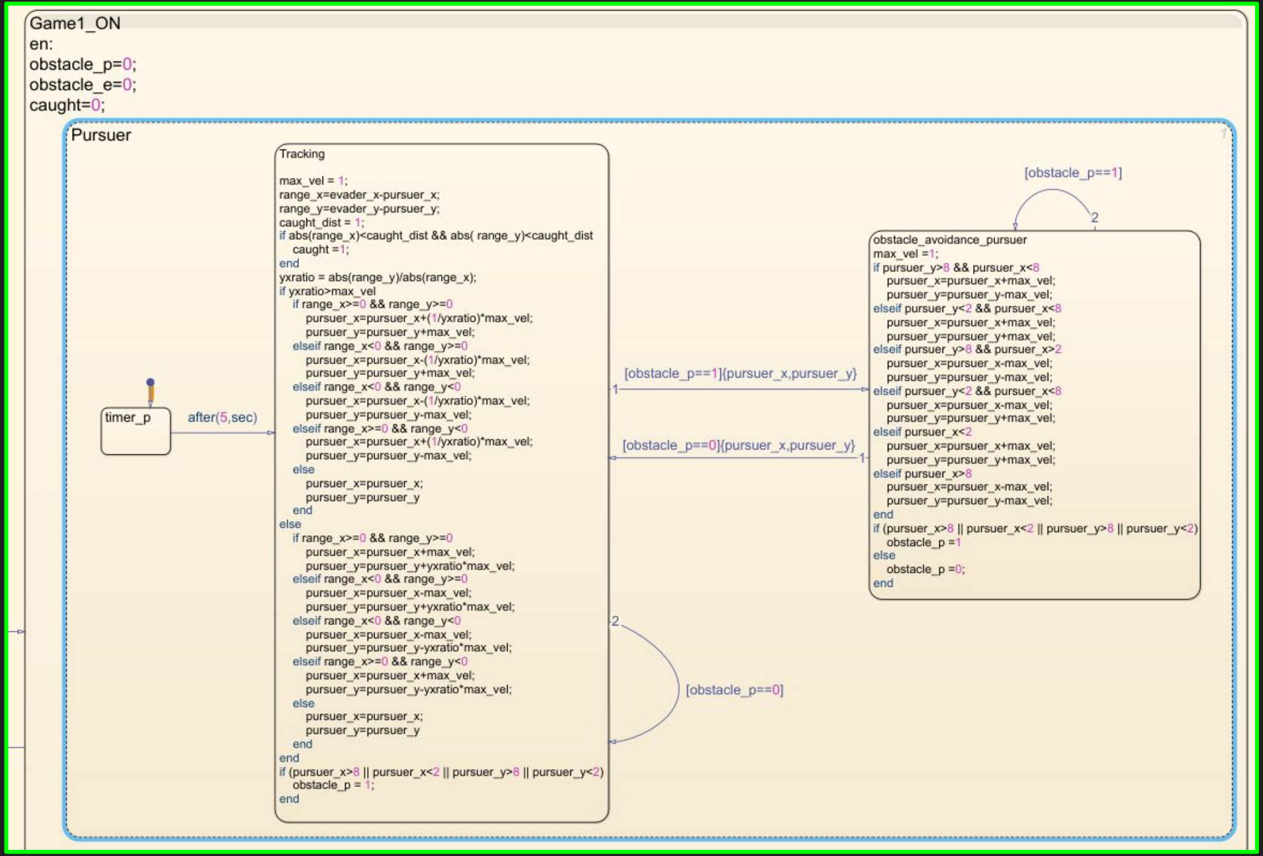




```
{caught=-1;
expired=0;
score_pursuer=0;
score_evader=0;
pursuer_x=3;
pursuer_y=3;
evader_x=7;
evader_y=7;}
```

```
if (caught==1)
   score_pursuer=score_pursuer+1
elseif (caught==0)
   score_evader=score_evader+1
end
caught=-1;
```

State Diagram and Algorithm



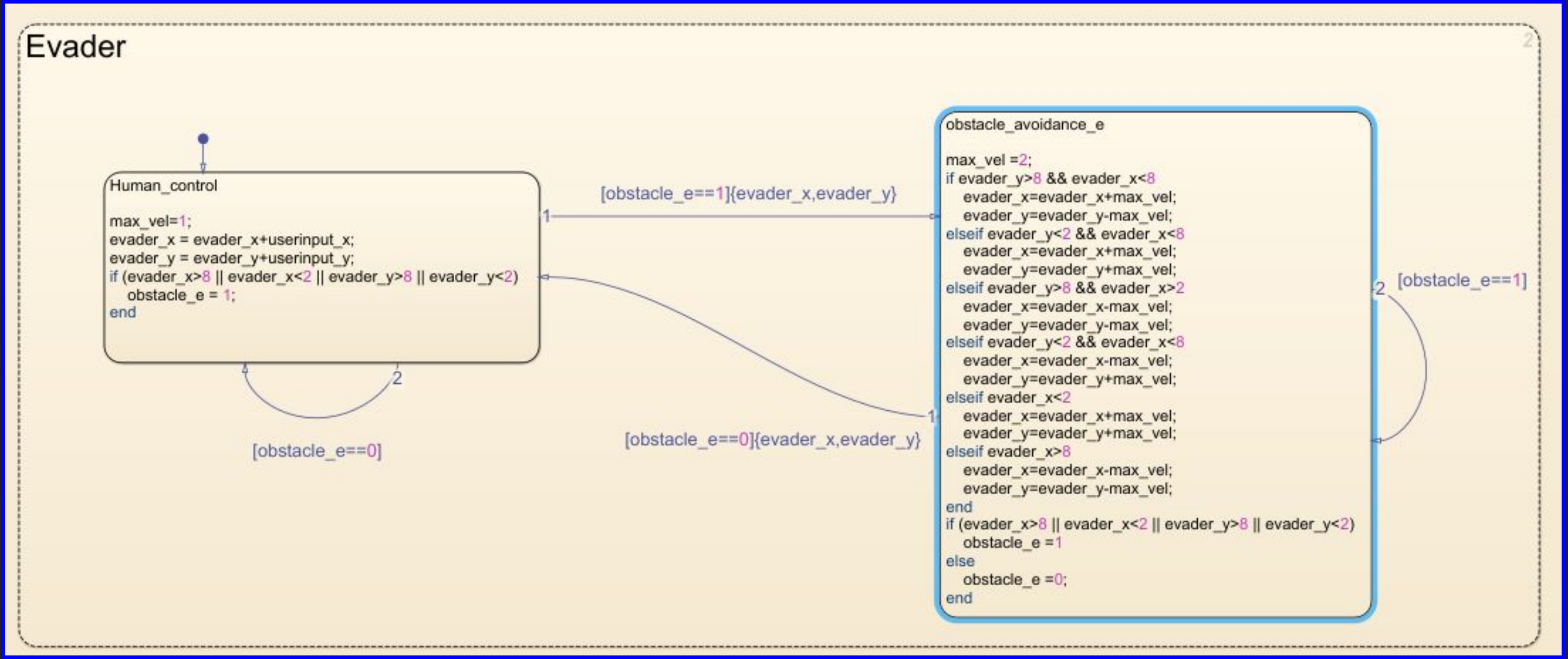
Game1_ON
Pursuer
timer_p
after(5,sec)
Tracking
obstacle_avoidance_pursuer
[obstacle_p==1]{pursuer_x,pursuer_y}
1
[obstacle_p==0]{pursuer_x,pursuer_y}
1
[obstacle_p==0]
2
[obstacle_p==1]
2

```
Game1_ON
en:
obstacle_p=0;
obstacle_e=0;
caught=0;
```

```
Tracking

max_vel = 1;
range_x=evader_x-pursuer_x;
range_y=evader_y-pursuer_y;
caught_dist = 1;
if abs(range_x)<caught_dist && abs( range_y)<caught_dist
   caught =1;
end
yxratio = abs(range_y)/abs(range_x);
if yxratio>max_vel
   if range_x>=0 && range_y>=0
      pursuer_x=pursuer_x+(1/yxratio)*max_vel;
      pursuer_y=pursuer_y+max_vel;
   elseif range_x<0 && range_y>=0
      pursuer_x=pursuer_x-(1/yxratio)*max_vel;
      pursuer_y=pursuer_y+max_vel;
   elseif range_x<0 && range_y<0
      pursuer_x=pursuer_x-(1/yxratio)*max_vel;
      pursuer_y=pursuer_y-max_vel;
   elseif range_x>=0 && range_y<0
      pursuer_x=pursuer_x+(1/yxratio)*max_vel;
      pursuer_y=pursuer_y-max_vel;
   else
      pursuer_x=pursuer_x;
      pursuer_y=pursuer_y
   end
else
   if range_x>=0 && range_y>=0
      pursuer_x=pursuer_x+max_vel;
      pursuer_y=pursuer_y+yxratio*max_vel;
   elseif range_x<0 && range_y>=0
      pursuer_x=pursuer_x-max_vel;
      pursuer_y=pursuer_y+yxratio*max_vel;
   elseif range_x<0 && range_y<0
      pursuer_x=pursuer_x-max_vel;
      pursuer_y=pursuer_y-yxratio*max_vel;
   elseif range_x>=0 && range_y<0
      pursuer_x=pursuer_x+max_vel;
      pursuer_y=pursuer_y-yxratio*max_vel;
   else
      pursuer_x=pursuer_x;
      pursuer_y=pursuer_y
   end
end
if (pursuer_x>8 || pursuer_x<2 || pursuer_y>8 || pursuer_y<2)
   obstacle_p = 1;
end
```

```
obstacle_avoidance_pursuer
max_vel =1;
if pursuer_y>8 && pursuer_x<8
   pursuer_x=pursuer_x+max_vel;
   pursuer_y=pursuer_y-max_vel;
elseif pursuer_y<2 && pursuer_x<8
   pursuer_x=pursuer_x+max_vel;
   pursuer_y=pursuer_y+max_vel;
elseif pursuer_y>8 && pursuer_x>2
   pursuer_x=pursuer_x-max_vel;
   pursuer_y=pursuer_y-max_vel;
elseif pursuer_y<2 && pursuer_x<8
   pursuer_x=pursuer_x-max_vel;
   pursuer_y=pursuer_y+max_vel;
elseif pursuer_x<2
   pursuer_x=pursuer_x+max_vel;
   pursuer_y=pursuer_y+max_vel;
elseif pursuer_x>8
   pursuer_x=pursuer_x-max_vel;
   pursuer_y=pursuer_y-max_vel;
end
if (pursuer_x>8 || pursuer_x<2 || pursuer_y>8 || pursuer_y<2)
   obstacle_p =1
else
   obstacle_p =0;
end
```

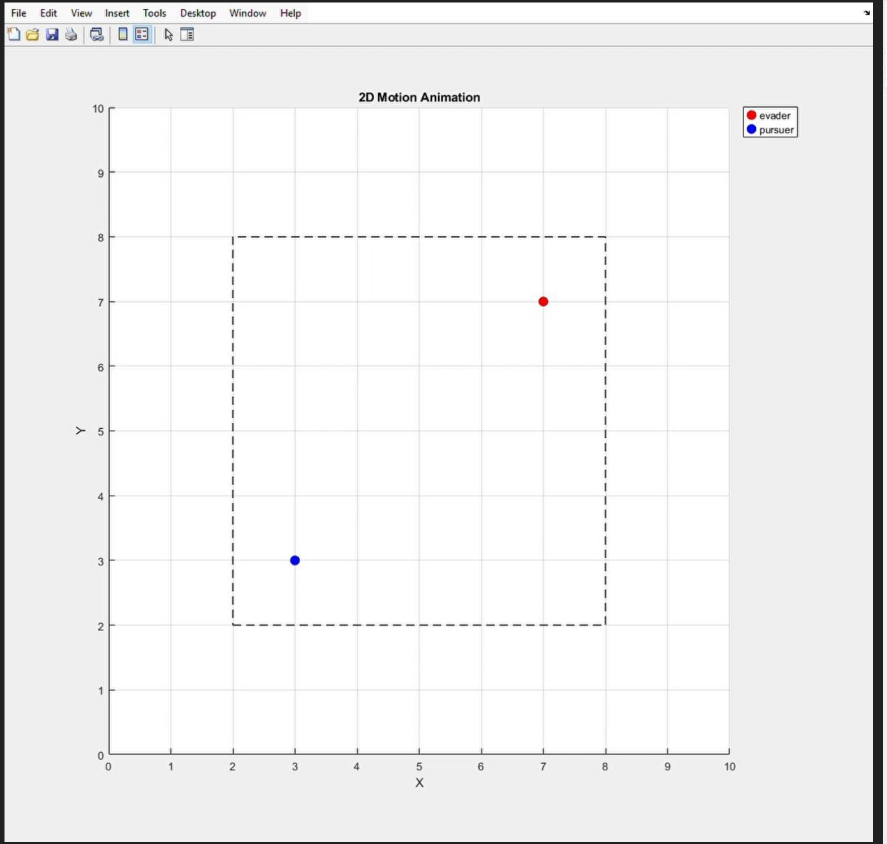
State Diagram and Algorithm

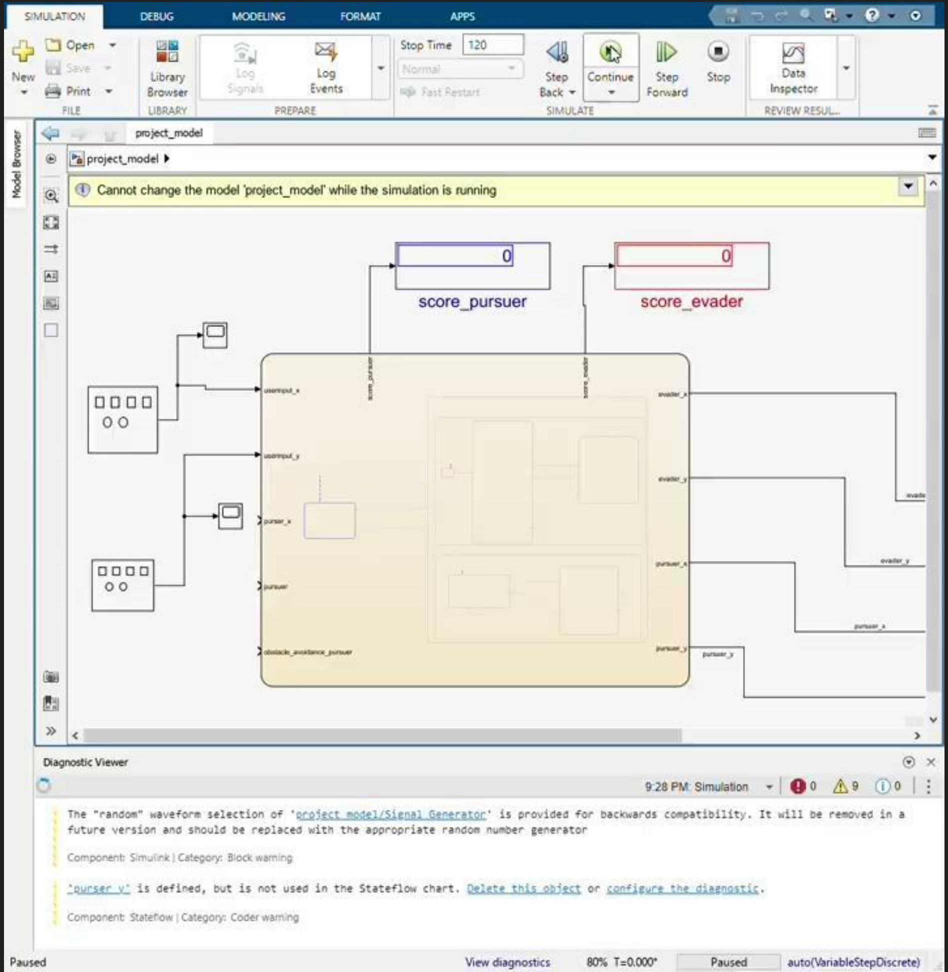

Evader
2
1
[obstacle_e==1]{evader_x,evader_y}
2
[obstacle_e==0]
1
[obstacle_e==0]{evader_x,evader_y}
2
[obstacle_e==1]

```
Human_control

max_vel=1;
evader_x = evader_x+userinput_x;
evader_y = evader_y+userinput_y;
if (evader_x>8 || evader_x<2 || evader_y>8 || evader_y<2)
    obstacle_e = 1;
end
```

```
obstacle_avoidance_e

max_vel =2;
if evader_y>8 && evader_x<8
   evader_x=evader_x+max_vel;
   evader_y=evader_y-max_vel;
elseif evader_y<2 && evader_x<8
   evader_x=evader_x+max_vel;
   evader_y=evader_y+max_vel;
elseif evader_y>8 && evader_x>2
   evader_x=evader_x-max_vel;
   evader_y=evader_y-max_vel;
elseif evader_y<2 && evader_x<8
   evader_x=evader_x-max_vel;
   evader_y=evader_y+max_vel;
elseif evader_x<2
   evader_x=evader_x+max_vel;
   evader_y=evader_y+max_vel;
elseif evader_x>8
   evader_x=evader_x-max_vel;
   evader_y=evader_y-max_vel;
end
if (evader_x>8 || evader_x<2 || evader_y>8 || evader_y<2)
   obstacle_e =1
else
   obstacle_e =0;
end
```

Simulation





Cannot change the model 'project_model' while the simulation is running

```
The "random" waveform selection of 'project_model/Signal Generator' is provided for backwards compatibility. It will be removed in a
future version and should be replaced with the appropriate random number generator
```

Component: Simulink | Category: Block warning

```
'purser_y' is defined, but is not used in the Stateflow chart. Delete this object or configure the diagnostic.
```

Component: Stateflow | Category: Coder warning

Key Applications from Lecture

- ROS (Robot Operating System):
  - Nodes, topics, subscribers, pursuers, etc.
- Hybrid Systems
  - Robot has both continuous and discrete behavior
  - Continuous: robot controlled by 30s game timer
  - Discrete: when target is detected, instantly jump from searching to pursuing state
- Scheduling & Latency Management
  - rospy.rate()
  - Experiment with adjusting rate to improve accuracy of pursuer robot
  - Remove unnecessary calculations from the loop
- Timed Interrupts
  - 30s game timer interrupts operation of both robots to signify end of the game
- Hierarchical State Machine with Reset Transition
  - From the pursuit state, robot can enter captured state
- Dealing with Sensor Errors
  - Implemented filter on pursuer using running average depth image measurement of 5 images
- I/O Polling
  - LiDAR, depth image, HSV camera image update every ROS loop

Specifications

- Yahboom controlled by human pilot with keyboard
  - Lidar obstacle avoidance overtakes human controller when within specified collision range
  - Buzzer sounds when obstacle avoidance takes control
- Yahboom autonomously controlled by onboard tracking algorithm
  - Lidar obstacle avoidance initiates reverse command when within 0.5m of an obstacle
  - Robot will spin to search for target when nothing detected
  - Autonomous robot will track and pursue the evader robot when detected, specifically looking for the color red
  - LED light strip displays unique color corresponding to state of the pursuit (i.e. searching, tracking, captured, waiting)
- Within the enclosed arena
  - Autonomous Yahboom pursues human controlled Yahboom
  - When autonomous robot captures evader robot, autonomous robot pauses for 10s
  - When autonomous robot captures evader robot, timer resets and win counter ticks up one point for the pursuer
  - When human-controlled robot evades autonomous robot for 30 seconds, timer resets and and win counter ticks up one point for the evader

Technical Challenges

Multi-agent Communication

- Defining namespaces and preventing topic and node clashes
- Public network vs. local hotspot on boot up

Latency Issues for Search and Track Algorithm Causes Latency Pursuer Velocity Updates

- Caused by computation time and bandwidth required to SSH into control
- SSH incompatibility with opencv
- **Remedy**: eliminate data-intensive feedback such as the live video and depth camera stream

Control of Evader Robot

- Joystick controller disconnects frequently and was generally unreliable
- Latency with SSH and keyboard control

Color Tracking Detects Unwanted Objects in the Environment

- **Remedy:** Extremely fine tuning of the HSV color detection range
- **Remedy:** Reduce FOV of pursuer to only where it will see the evader

Division of Labor

- Alex Melnick - Co-developed pursuer-bot, especially color tracking and LiDAR integration, full integration
- Maura Mulligan - Evader robot, gameplay, full integration
- Megha Shah - Evader robot, simulation, full integration
- Mitch Hornak - Developed pursuer robot, full integration

Live Demonstration

Pursuer Light Bar State Key:

Yellow: Searching State

Red: Pursuit State

Blue: Evader Captured State

Green: LiDAR Safety State

Purple: Game Over State