Sensor Coverage Simulation Code Version Tracking:

Version 0.1:

Initial development of sensor.py, target.py, environment.py, and env\_test.py

Sensors and targets are successfully generated.

Sensors track targets as they enter their fov and until they exit.

Battery level deplete as the sensors are active.

Env\_test displays targets that were tracked.

1. Sensor.py
   1. Developed the base class
   2. Functions
      1. draw sensors
      2. update\_sensor\_fov
      3. update\_energy
2. target.py
   1. Developed the base class
   2. Functions
      1. Draw\_target
      2. Move
3. Environment.py
   1. Developed base class
   2. Functions
      1. Create\_env
      2. Generate\_target\_list
      3. Run\_env
4. Env\_testing.py
   1. Initial testing program

Version 0.2: 8/4 – 8/10

Sensor.py

Changed self.active to self.mode

The options for mode are idle, sleep, wake\_up, and active. These are strings

Updated all instances of self.active to self.mode =’XXXX’ as required

Update\_energy function

Logic for the different level of energy consumption. If the energy level drop to, or below zero, the sensor is placed into sleep mode and the energy level is set to 0.

Draw\_sensors function

Created a different skin for when the sensor is in idle. A transparent gray is used.

Allows for the FOV to be viewed in idle and active mode

If mode is sleep then the sensor will turn completely gray without a FOV visible.

Update\_sensor\_fov function

Added method. Default is None. The method allows for either direct assignment if ‘directed’ or auto updating to tracking a target if in FOV if set to NONE

Environment.py

Added self.energy and self.energy\_total to the class

Create \_env

Updated the total initial energy as the sensors are created

Run\_env

Track total available energy from all sensors in each frame with current\_energy and append the total to the self.energy

Env\_stats

Processes the list of tracked items and returns an np array

Provides the unique number of targets tracked and their ids

Tracks the energy consumption by time step

Has an option for plotting the targets tracked and total available energy

Returns the tracked and energy data

Created env\_testing\_notebook.ipynb

Pushed the env\_testing.py into a jupyter notebook to allow for manipulating the data generated in the environment.

Next efforts:

Generate a batch of sensors based on locations and initial fov angle.

Implement the dt to limit framerate?

Develop a function to calc and capture all stats from the environment instance for later analysis?

Coverage percent per time tick?

Reset function for sensors that go inactive -reset their initial angle

Add id to sensors