

# MP2: Ancient Mine Exploration

---

By: Rohan Tabish

# What have we learned so far ?



# Recap - Asimov Laws for Robotics



1. A robot may not injure a human being or, through inaction, allow a human being to come to harm

# Recap - Asimov Laws for Robotics



1. A robot may not injure a human being or, through inaction, allow a human being to come to harm



2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law

# Recap - Asimov Laws for Robotics



1. A robot may not injure a human being or, through inaction, allow a human being to come to harm



2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law



3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws

# What kinds of robots we want?



# What kinds of robots we want?



# What kinds of robots we want?



Let us put together what we have learned so far into practice in our next machine problem.





# MP2 - Ancient Mine Exploration



Farmer planting tree

# MP2 - Ancient Mine Exploration



Farmer planting tree



Uncovering a maze

# MP2 - Ancient Mine Exploration



Farmer planting tree



Uncovers a maze



A research team is assembled to send a robot

# MP2 - Ancient Mine Exploration



Farmer planting tree



Uncovers a maze



A research team is assembled to send a robot

The maze is labeled as **archeological** site.  
So there are **requirements** that robot must follow.



# Requirements

- Requirement # 1: Look but do not touch
  - The robot should not run into or damage any object
- Requirement # 2: Protect yourself
  - The robot should avoid self-damage because there are cliffs and traps
- Requirement # 3: Run some analytics
  - Scientist wants to download analytics tasks to the robot that show run while exploring the maze.  
The task runs analytics on the data captured by the robot.
- Requirement # 4: Finish as quickly as possible

# Translating into Requirements Engineering



- **Safety-critical requirements**
  - What are the hazards from environment?



- **Mission-critical requirements**
  - What needs to be done for this mission to be successful?



- **Performance-critical requirements**
  - What decides how well the mission is executed?

# Translating into Requirements Engineering



- **Safety-critical:** basic survival/safety



- **Mission-critical:** the purpose of the mission



- **Performance-critical:** attributes describing how well the mission is performed

# Translating into Requirements Engineering



- **Safety-critical: basic survival/safety**
  - Stop when hitting the wall, wheels dropping, cliff signal, overcurrent



- **Mission-critical: the purpose of the mission**



- **Performance-critical: attributes describing how well the mission is performed**



# Translating into Requirements Engineering



- **Safety-critical: basic survival/safety**
  - Stop when hitting the wall, wheels dropping, cliff signal, overcurrent



- **Mission-critical: the purpose of the mission**
  - Violating them causes the mission to fail, but the system will remain unharmed
  - Track trajectory, take pictures, run external task



- **Performance-critical: attributes describing how well the mission is performed**

# Translating into Requirements Engineering



- **Safety-critical: basic survival/safety**
  - Stop when hitting the wall, wheels dropping, cliff signal, overcurrent



- **Mission-critical: the purpose of the mission**
  - Violating them causes the mission to fail, but the system will remain unharmed
  - Track trajectory, take pictures, run external task



- **Performance-critical: attributes describing how well the mission is performed**
  - Travel time, time to finish entire mission (including processing pictures), photo analysis

# How many threads we need ?

- Wall sensor
- Cliff/wheel drop sensor
- Overcurrent sensor
- Motion control
- Sound/LED
- Tracking trajectory
- Taking photos
- Matching pictures

# Map the Tasks to Criticality Levels We Defined



- **Safety-critical: Basic Survival/Safety**
  - Highest Priority
  - Least Priority, fastest response
  - Checking safety-related sensors, stop immediately if needed



- **Mission-critical: the purpose of the mission**
  - Second Highest Priority
  - Motion Control, Tracking and Navigation etc
  - Follow the wall, tracking trajectory



- **Performance-critical: Quality of the mission**
  - Lowest Priority
  - Computer Vision
  - Taking pictures, run openCV

# Map the Tasks to Criticality Levels We Learned



- **Safety-critical: Basic Survival/Safety**

We atleast need three threads.



- **Mission-critical: the purpose of the mission**
  - Second Highest Priority
  - Motion Control, Tracking and Navigation etc
  - Follow the wall, tracking trajectory



- **Performance-critical: Quality of the mission**
  - Lowest Priority
  - Computer Vision
  - Taking pictures, run openCV

# Map the Tasks to Criticality Levels We Learned



- **Safety-critical: Basic Survival/Safety**

We atleast need three threads.



- **Mission-critical: the purpose of the mission**
  - Second Highest Priority
  - Motion Control, Tracking and Navigation etc
  - Follow the wall, tracking trajectory

Can we have more threads? **Sure.**

# Map the Tasks to Criticality Levels We Learned



- **Safety-critical: Basic Survival/Safety**

We atleast need three threads.



- **Mission-critical: the purpose of the mission**
  - Second Highest Priority
  - Motion Control, Tracking and Navigation etc
  - Follow the wall, tracking trajectory

Can we have more threads? **Sure.**

**Caution: Having too many threads can introduce overhead.**

# Alright, I can do this with three threads?



- **Safety-critical: Basic Survival/Safety**
  - Highest Priority
  - Least Priority, fastest response
  - Checking safety-related sensors, stop immediately if needed



- **Mission-critical: the purpose of the mission**
  - Second Highest Priority
  - Motion Control, Tracking and Navigation etc
  - Follow the wall, tracking trajectory



- **Performance-critical: Quality of the mission**
  - Lowest Priority
  - Computer Vision
  - Taking pictures, run openCV



# Alright, I can do this with three threads?



- **Safety-critical: Basic Survival/Safety**

- Highest Priority
- Least Priority, fastest response



**Wait: How do I assign priorities to threads?**

- Follow the wall, tracking trajectory



- **Performance-critical: Quality of the mission**

- Lowest Priority
- Computer Vision
- Taking pictures, run openCV

# Use sched\_priority to set the priority of threads?



```
pthread_attr_t attrSafety;  
sched_param paramSafety;  
pthread_attr_init (&attrSafety);  
pthread_attr_getschedparam (&attrSafety, &paramSafety);  
paramSafety.sched_priority = 4;  
pthread_attr_setschedparam (&attrSafety, &paramSafety);
```



```
pthread_attr_t attrMotion;  
sched_param paramMotion;  
pthread_attr_init (&attrMotion);  
pthread_attr_getschedparam (&attrMotion, &paramMotion);  
paramMotion.sched_priority = 3;  
pthread_attr_setschedparam (&attrMotion, &paramMotion);
```



```
pthread_attr_t attrVision;  
sched_param paramVision;  
pthread_attr_init (&attrVision);  
pthread_attr_getschedparam (&attrVision, &paramVision);  
paramVision.sched_priority = 2;  
pthread_attr_setschedparam (&attrVision, &paramVision);
```

# Use sched\_priority to set the priority of threads?



```
pthread_attr_t attrSafety;  
sched_param paramSafety;  
pthread_attr_init (&attrSafety);  
pthread_attr_getschedparam (&attrSafety, &paramSafety);  
paramSafety.sched_priority = 4;
```



Yay !! I can assign priorities now



```
pthread_attr_setschedparam (&attrMotion, &paramMotion);
```



```
pthread_attr_t attrVision;  
sched_param paramVision;  
pthread_attr_init (&attrVision);  
pthread_attr_getschedparam (&attrVision, &paramVision);  
paramVision.sched_priority = 2;  
pthread_attr_setschedparam (&attrVision, &paramVision);
```

# Use sched\_priority to set the priority of threads?



```
pthread_attr_t attrSafety;  
sched_param paramSafety;  
pthread_attr_init (&attrSafety);  
pthread_attr_getschedparam (&attrSafety, &paramSafety);  
paramSafety.sched_priority = 4;
```

What about periods of the tasks ?



```
pthread_attr_setschedparam (&attrMotion, &paramMotion);
```

```
pthread_attr_t attrVision;  
sched_param paramVision;  
pthread_attr_init (&attrVision);  
pthread_attr_getschedparam (&attrVision, &paramVision);  
paramVision.sched_priority = 2;  
pthread_attr_setschedparam (&attrVision, &paramVision);
```

# More on threading

- Put threads to sleep so that other threads can also run.
  - High priority thread should sleep less (e.g. 100 ms)
  - Low priority thread should sleep more (e.g. 2000 ms)
  
- The external thread runs separately so sleeping gives CPU to that thread to run.

# Other Issues to Keep in Mind

- Use mutex between threads when using shared resource such as serial port
  - Release mutex before going to sleep
  
- For MP2 you can take pictures inside the maze and analyze them outside the maze to save travel time.
  
- Find a sweet timing spot to take pictures
  - Too frequent will increase travel time
  - Too rare, you might not capture all the objects

# Other Issues to Keep in Mind

- Overcurrent sensor may give false positives
  - You might want to average or take median of values depending upon the window size
  
- Wall sensor
  - It has readings only when almost parallel to wall, maximum value and angle differ at different distance
  - What to do when signal reading is lost?
  
- More details can be found in the MP2 description.

Our Maze looks like this





# Other Points to Ponder

- We have a multicore processor - Can an attacker launch some attack and hog the bus from another core ?

Thank you  
Q & A

# Logistics

- Office Hours - Tuesday and Thursday 1:00 pm - 2:00pm - SC 2325 and SC 2327
- Q & A Piazza and office hours
- Email: [rtabish@illinois.edu](mailto:rtabish@illinois.edu)
- 10/31 at 11:59:59 pm
- Demo using signup sheet which gets posted on piazza after the deadline of the MP.