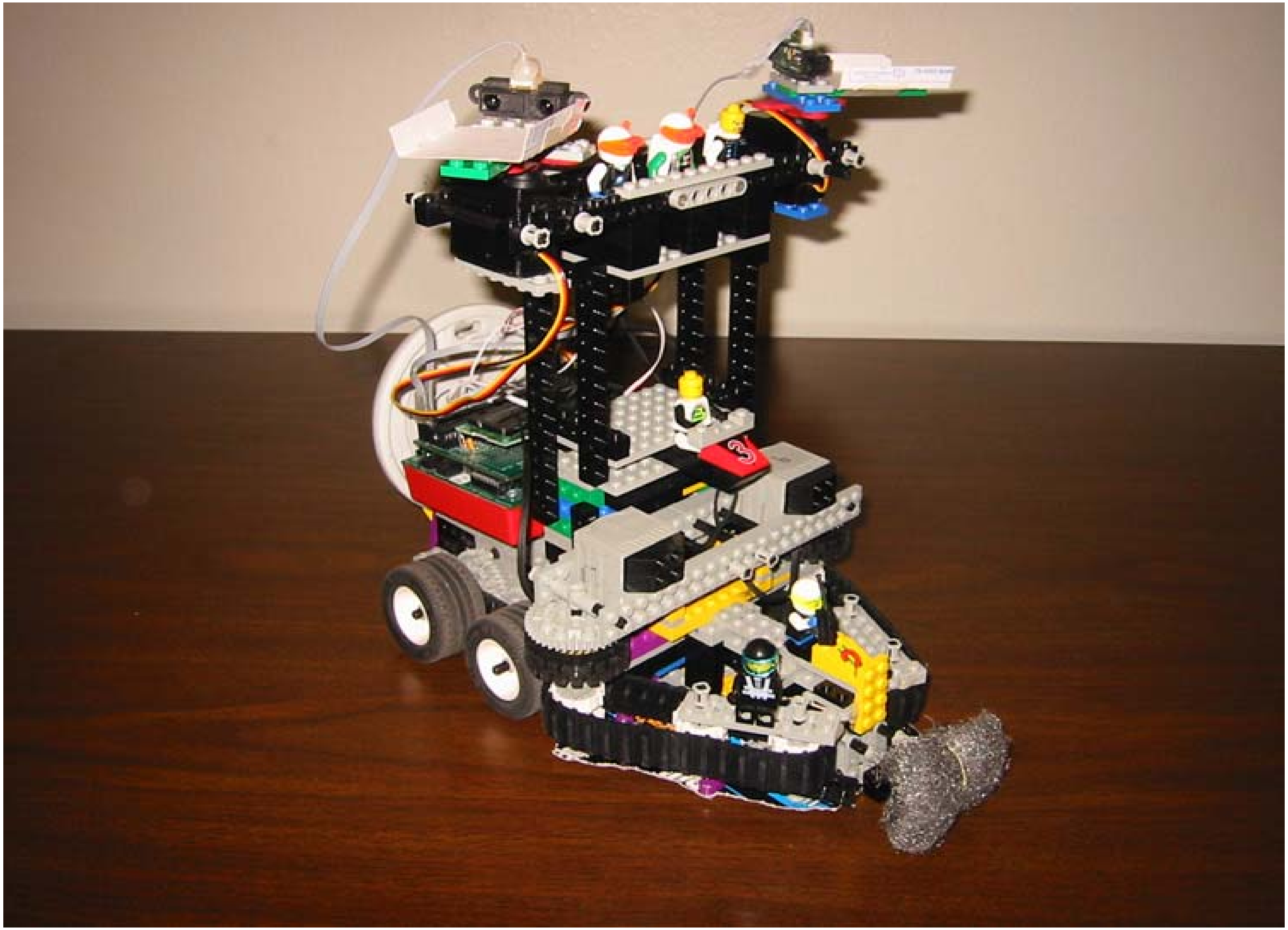
Robot Project 2: Presentation

Group 7:
Robert Moe
John Zumwalt
Mark Woehrer
Celi Sun

Robot Overview



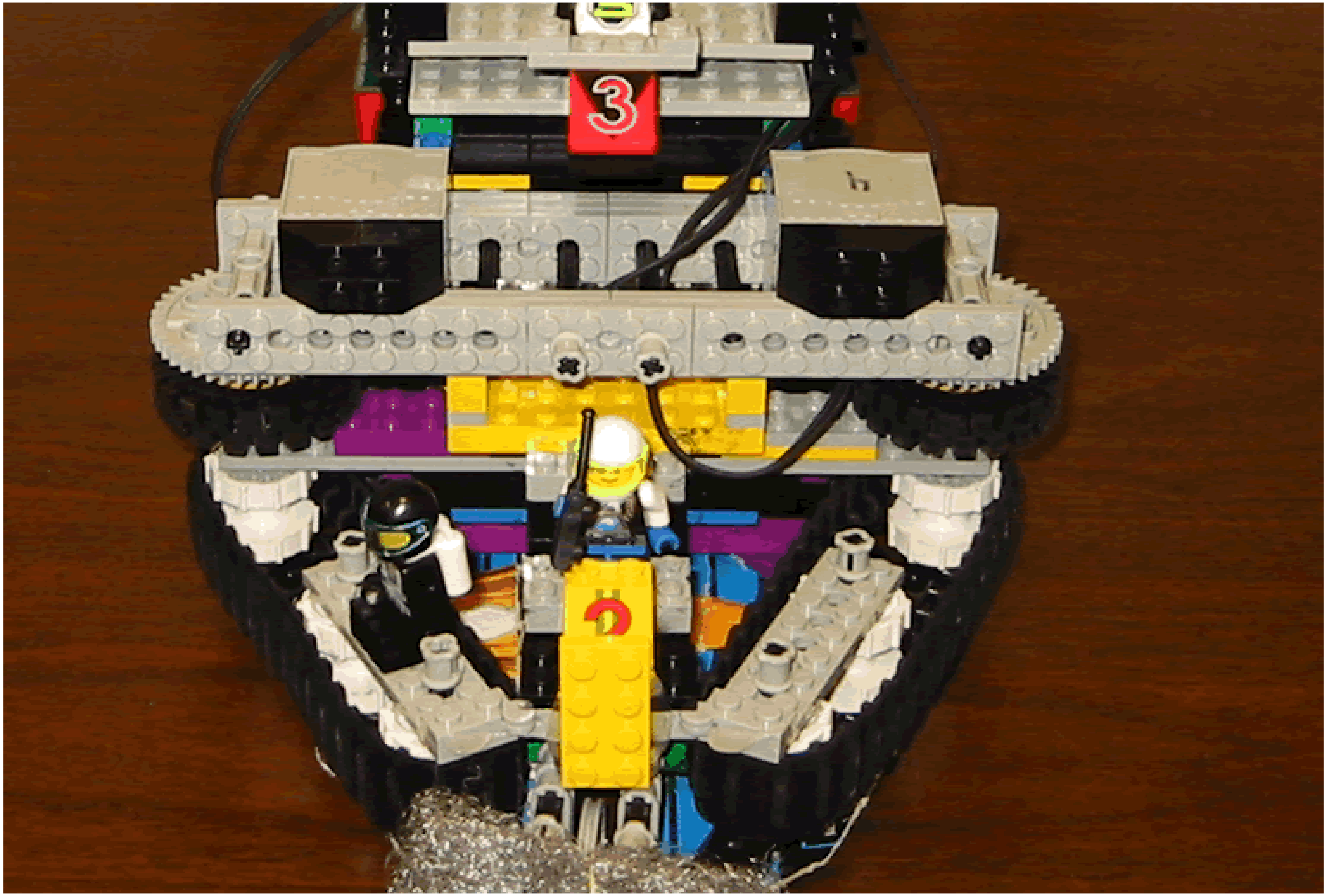
Robot Hardware Design

- Original Plan
  - Small design.
  - Avoid everything except light fixtures.
  - Sweeping lower range finder.
- Final Design
  - Very large design.
  - Avoid just the buckets.
  - Lower powered wedge for rocks.

Robot Hardware Design

- Powered Wedge
- Features
  - Powered 2 independent motors.
  - Tank Tracks to push rocks out of the way.
  - Push bumper on front to detect head on collisions.
  - Wheels on side to help the robot move along the wall.

Robot Powered Wedge



Robot Hardware Design

- Powered Wedge Advantages.
  - Useful to help to maneuver itself through tough rock obstacles.
  - No need to sense the rocks in the arena.
    - Easier code
  - Tank treads would pull robot out and around tight areas.

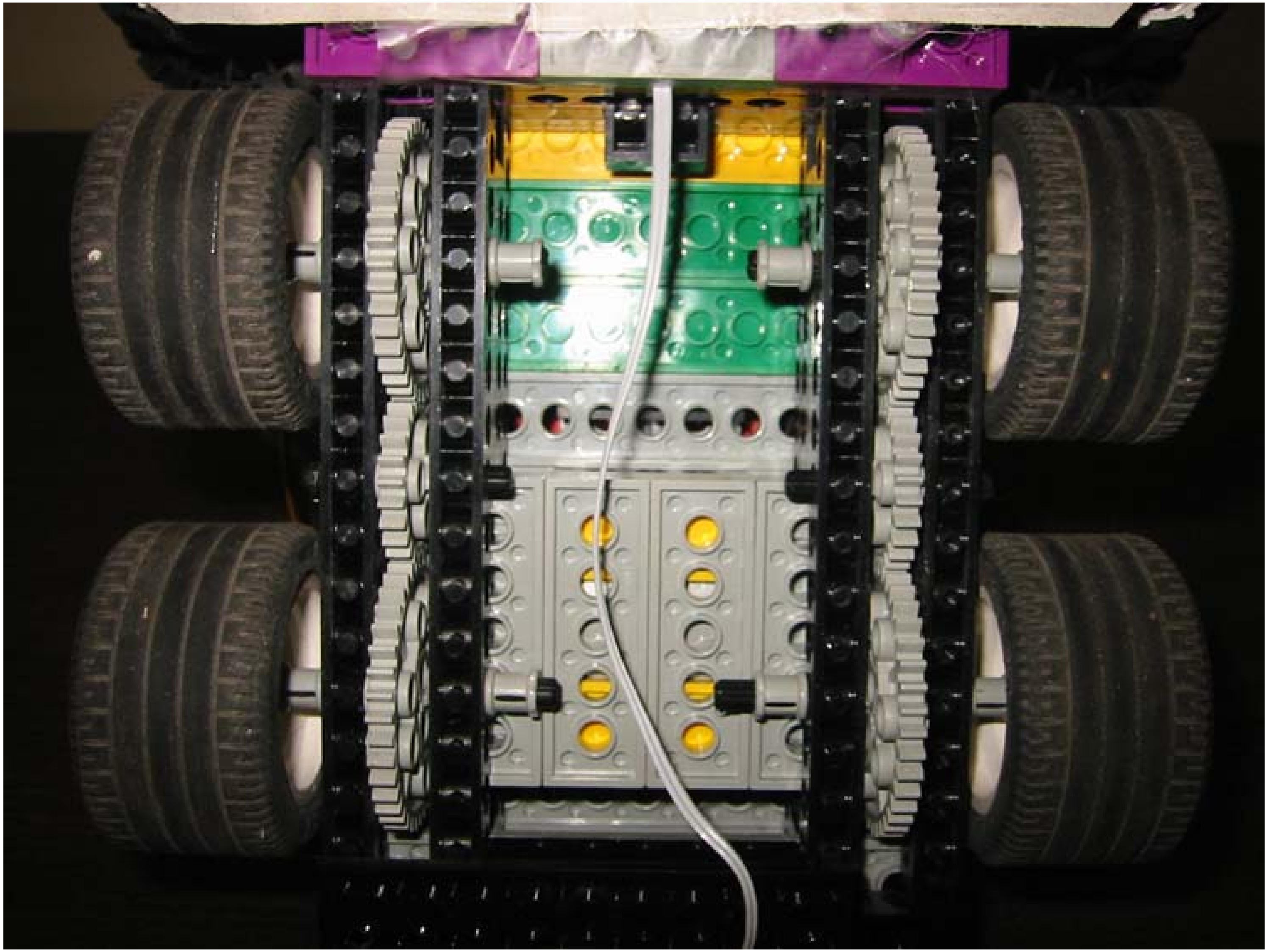
Robot Hardware Design

- Powered Wedge Disadvantages
  - Made the robot way to big.
    - Would get stuck in the corner of the arena.
  - Got stuck on the power cord during 1st run.
  - Harder to avoid buckets with long wedge sticking out.
  - Power consumption
    - Solved with tethered power source.

Robot Hardware Design

- Drive Train
  - □ Original Design.
    - Adder - Subtractor Design
      - □ Failed due to low torque on turns.
  - □ 4 wheel drive.
    - Extra power and torque to push rocks.
    - Easy to turn.
    - 2 independent drive motors for each side.

Robot Drive Train Design



Robot Hardware Design

- Sensors
  - Bump sensor
    - Detects bumps with the wall or large, heavy objects.
  - 2 Range Finders
    - Mounted on servos to sweep area for the tall obstacles.
    - Caused a lot of trouble when near a light source.

Robot Hardware Design

- Sensors cont'
  - 2 Light Sensors.
    - Mounted on each side
    - Used to align the robot towards the light using the difference between the 2 sensors.

Robot Software Design

- Original Design.
  - Using Subsumption Architecture
  - Each behavior having its own process.
- Final Design.
  - Used a state machine type architecture.
  - Robot would enter states depending on sensor readings

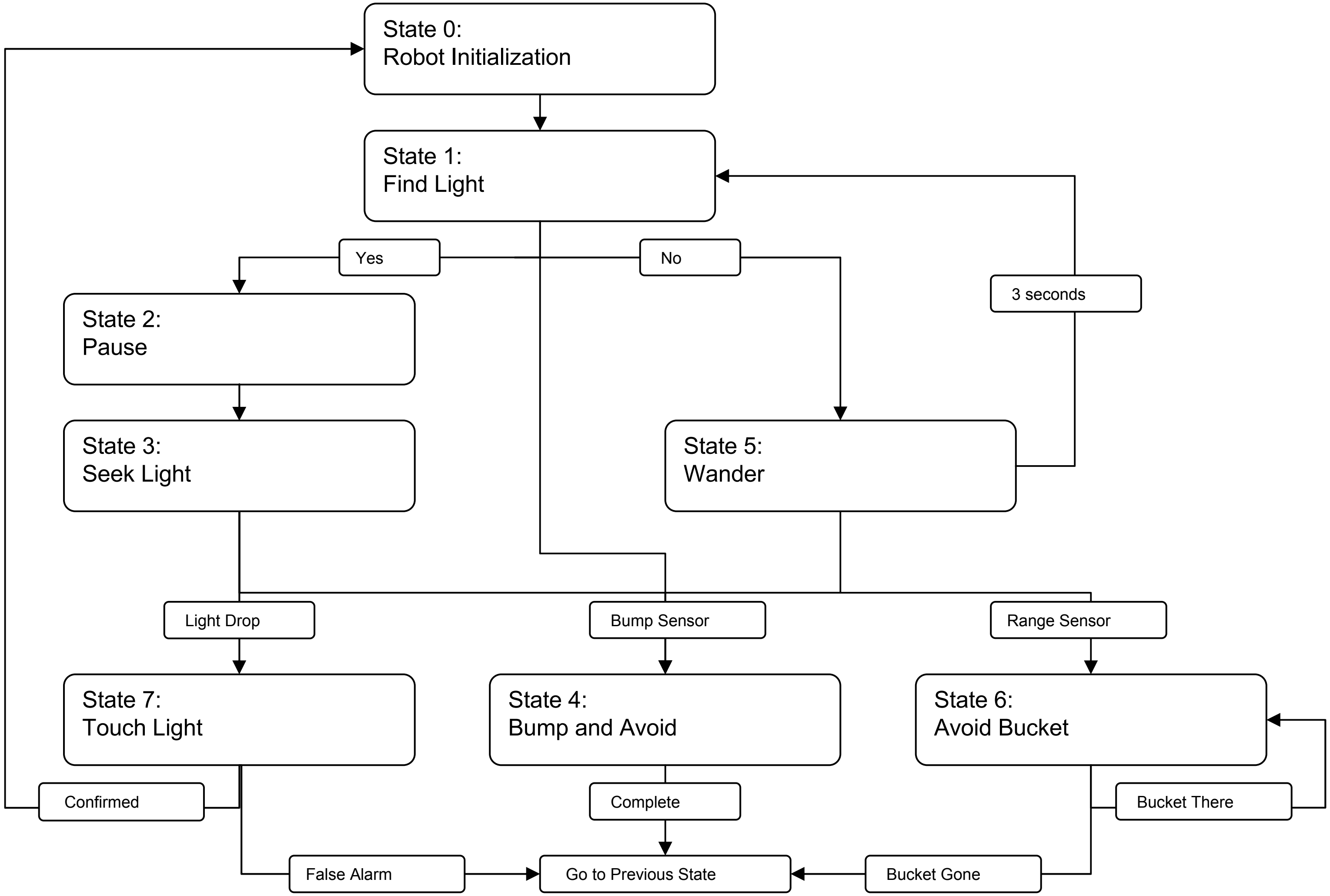
Robot Software Design

- State machine advantages.
  - No need for processes.
    - Had problems with processes working well on the handyboard.
  - Able to easily add a new state or branch for new behavior condition.
  - Easily debug which state the robot is in.

Robot Software Design

- State Machine Disadvantages
  - Code is hard to read and understand without knowledge of the way it works.
  - Code can become long, lots of outside variables to keep track of state.
  - No separation between behavior and hardware.

Robot Software Design



State 0:
Robot Initialization

State 1:
Find Light

Yes

No

3 seconds

State 2:
Pause

State 3:
Seek Light

State 5:
Wander

Light Drop

Bump Sensor

Range Sensor

State 7:
Touch Light

State 4:
Bump and Avoid

State 6:
Avoid Bucket

Confirmed

Complete

Bucket There

False Alarm

Go to Previous State

Bucket Gone

Conclusions

- Successes
  - Software found all the lights.
    - Would turn and drive towards every light.
  - Hardware was able to handle the rocks.
    - Robot moved rocks out of the way okay.
- Failures
  - Hardware was to bulky for arena.
    - Got caught on the corner or on taped cords.
  - Range sensors were flakey around light sources.
    - Would see things that were not there.

The End.

- Questions?