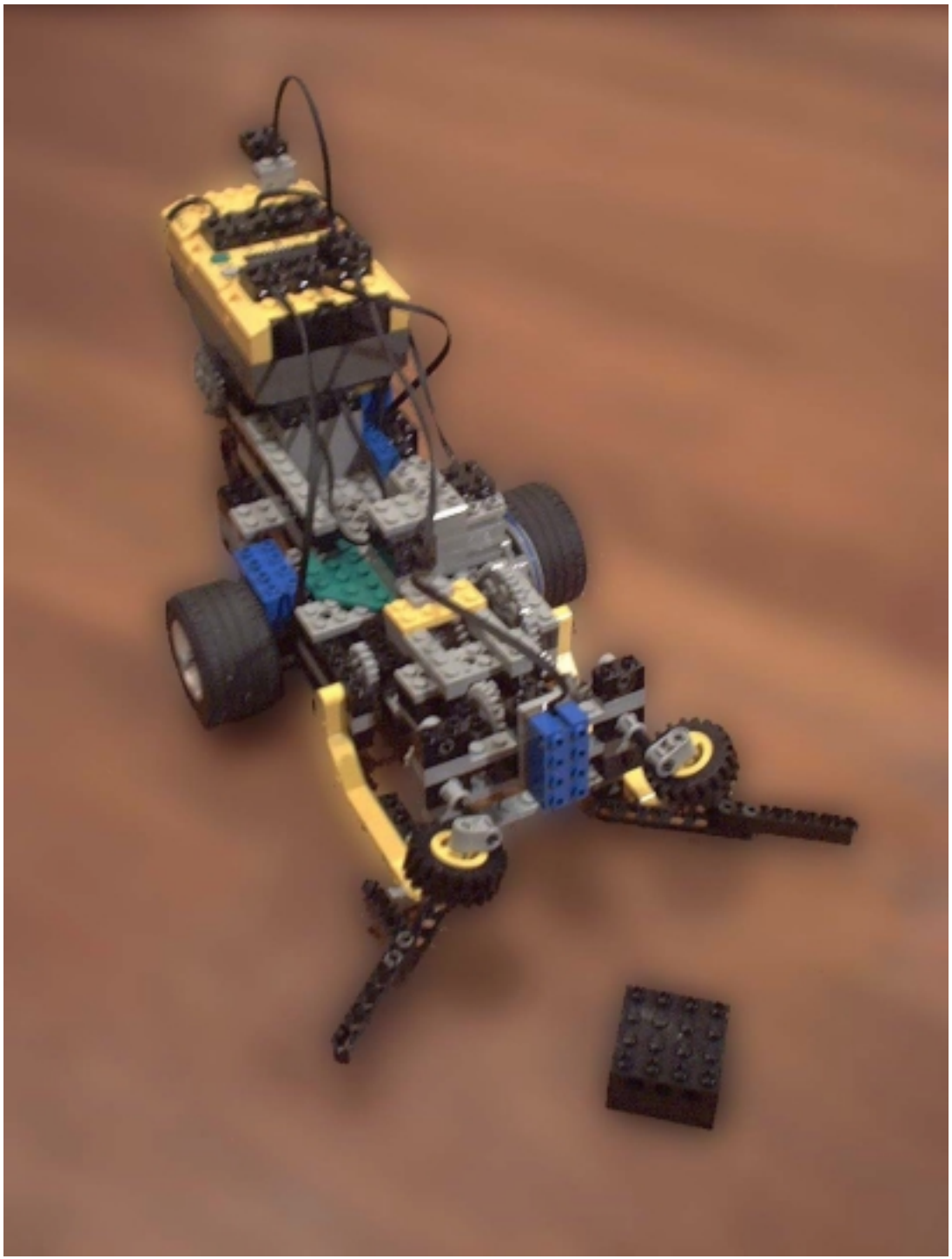
Search and Rescue Robot

Shannon Zelinski
Chris Cortopassi



Outline

- Problem Formation
- Implementation Goals
- Low Level Control (reusable commands)
- High Level Control (threading)
- Scheduling

Problem Formation

- Autonomous!
- Search for block within contained rectangular area
- Once found, pick up block, return it to starting point and orientation.
- Ability to pause anywhere in process via network linked RCX

Implementation Goals

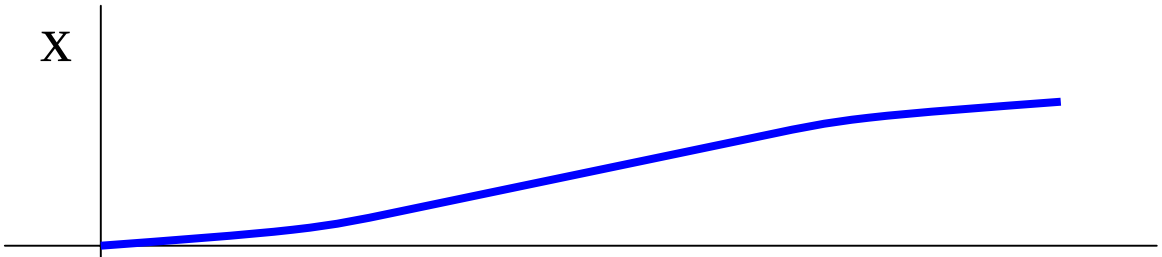
- Simplicity
  - Use minimal threads composed of sequential command blocks
  - Steer and drive separately
  - Fixed turning radius
  - Snake search pattern
- Accuracy
  - Motor profiling
  - Human calibration
  - Weight distribution

Low Level Control

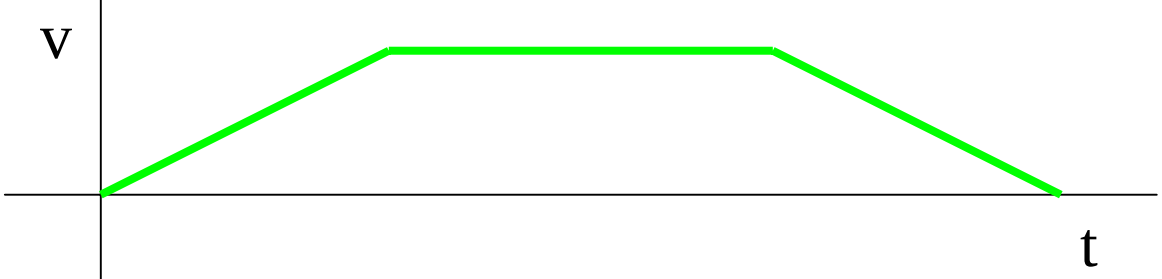
- Change steering
  - Steers to given absolute steering position
- Drive
  - Drives forward by the given relative distance
- Arm control
  - Opens and lifts arm or closes and lifts arm depending on the given direction

Velocity Profiling

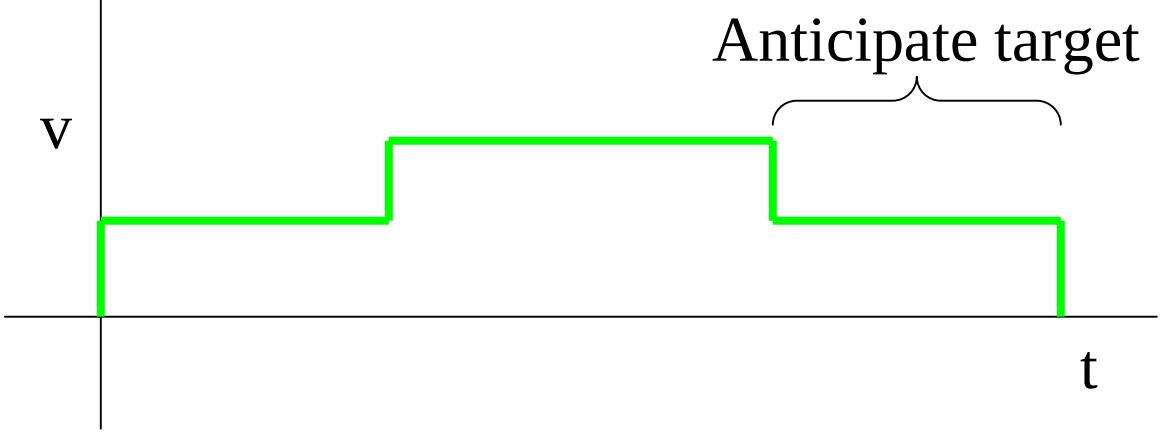
Ideal Smooth Position Profile



Ideal Trapezoidal Velocity Profile

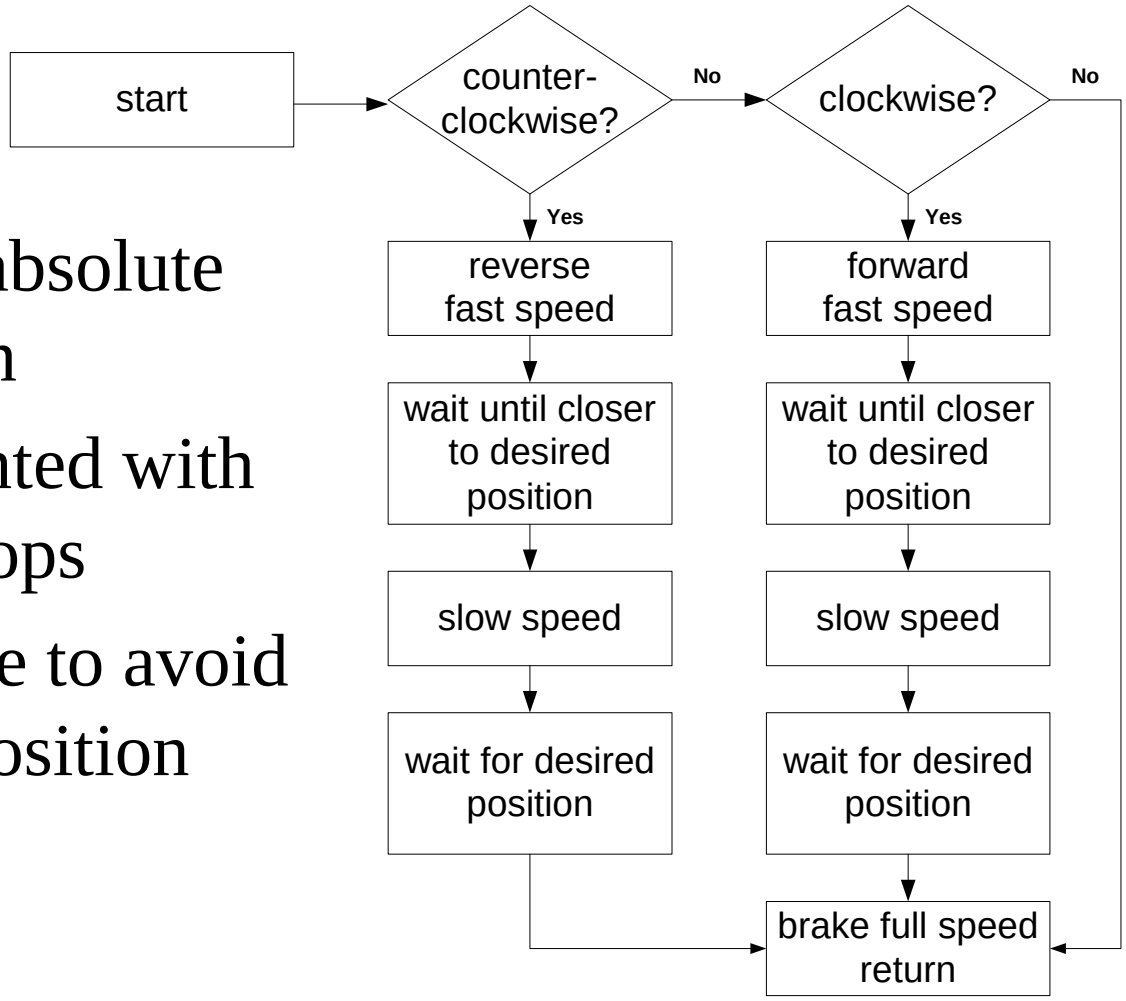


Our Simple Approximation



Change_steering(value)

- Value=desired absolute steering position
- Waits implemented with NULL while loops
- Slow when close to avoid over shooting position

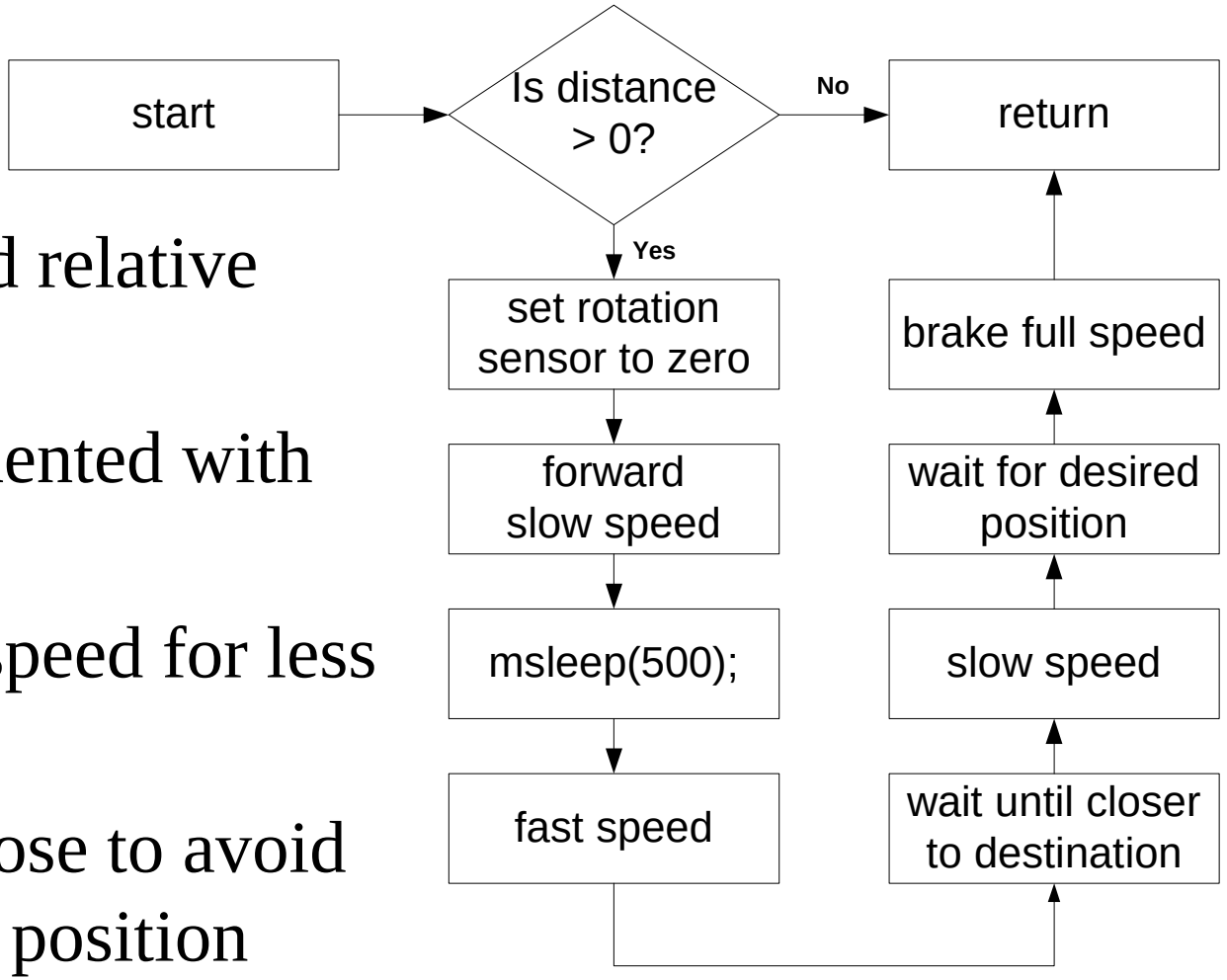


Steering_change code

```
void change_steering(int value)
{               // counterclockwise
  if (value < ROT_STEER_DATA)
  {
    motor_c_dir(rev);
    motor_c_speed(F_STEER_SPEED);
    while(ROT_STEER_DATA > value +
     OVER_SHOOT) {} //NULL while loop
    motor_c_speed(S_STEER_SPEED);
    while(ROT_STEER_DATA > value) {}
  }
                  // clockwise
else if (value > ROT_STEER_DATA)
  {
    motor_c_dir(fwd);
    motor_c_speed(F_STEER_SPEED);
    while(ROT_STEER_DATA < value -
     OVER_SHOOT) {}
    motor_c_speed(S_STEER_SPEED);
    while(ROT_STEER_DATA < value) {}
  }
  motor_c_dir(brake);    //full speed breaking
  motor_c_speed(255);
  msleep(SLEEP_TIME);
}
```

Drive(value)

- Value=desired relative distance
- Waits implemented with wait_event
- Start at slow speed for less wheel slip
- Slow when close to avoid over shooting position



Drive code

```
void drive(int distance)
{
    if(distance > 0)
    {       // set current position to zero
            ds_rotation_set(&ROT_DRIVE_SENSOR, 0);
            motor_b_dir(fwd);
            motor_b_speed(S_DRIVE_SPEED);
            msleep(500);
            motor_b_speed(F_DRIVE_SPEED);  // wakeup function used
            wait_event(&drive_wakeup, distance - OVER_SHOOT);
            motor_b_speed(S_DRIVE_SPEED);
            wait_event(&drive_wakeup, distance);
            motor_b_dir(brake);          // full speed breaking
            motor_b_speed(255);
            msleep(SLEEP_TIME);
    }
}
```

Arm(dir)

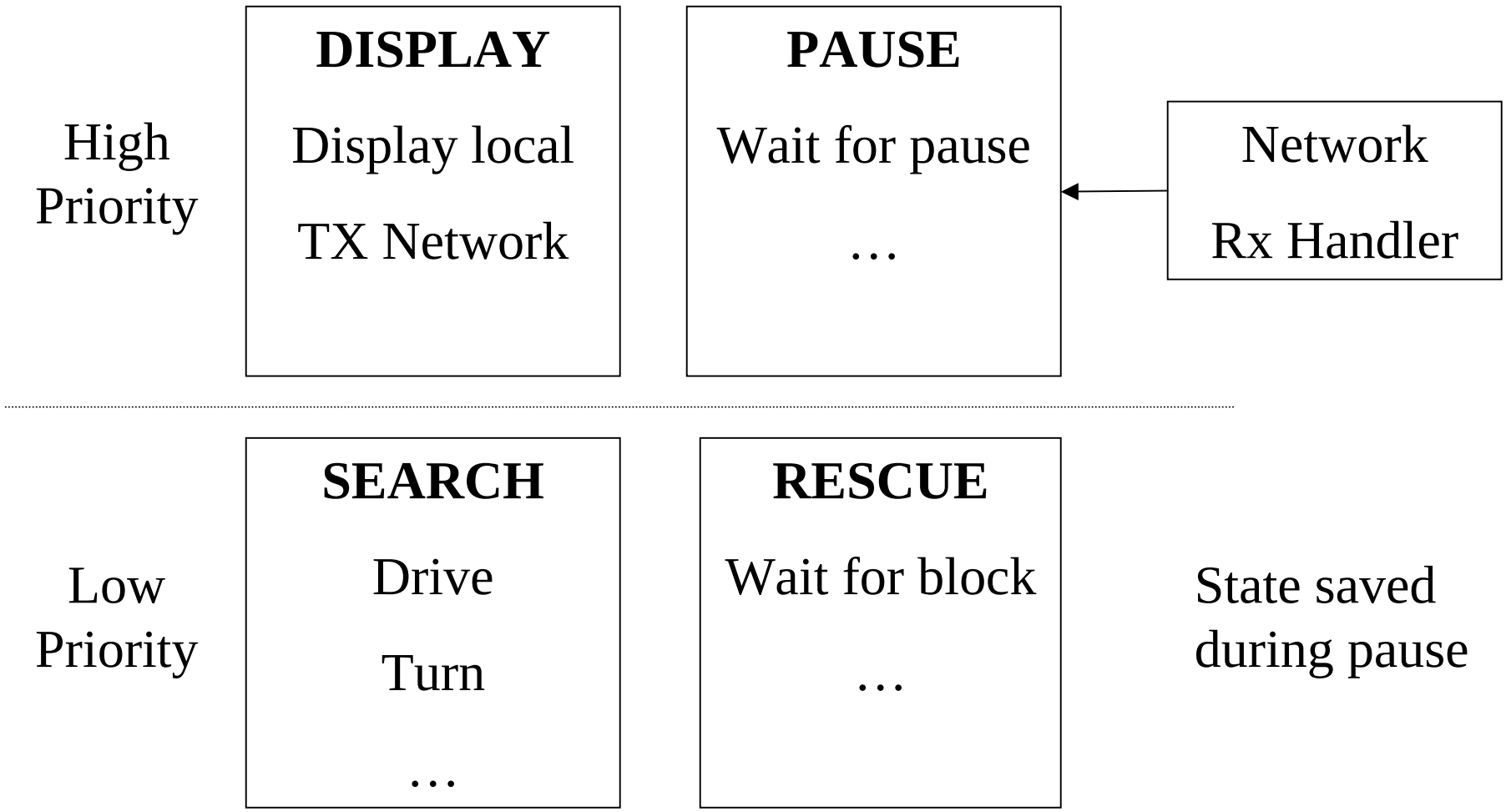
```
#define OPEN fwd          //arm(OPEN) opens arm
#define CLOSE rev         //arm(CLOSE) closes arm

void arm(short dir)
{
   motor_a_speed(ARM_SPEED);
   motor_a_dir(dir);
   wait_event(&touch_wakeup, TOUCHED);
   motor_a_speed(0);
   msleep(SLEEP_TIME);
}
```

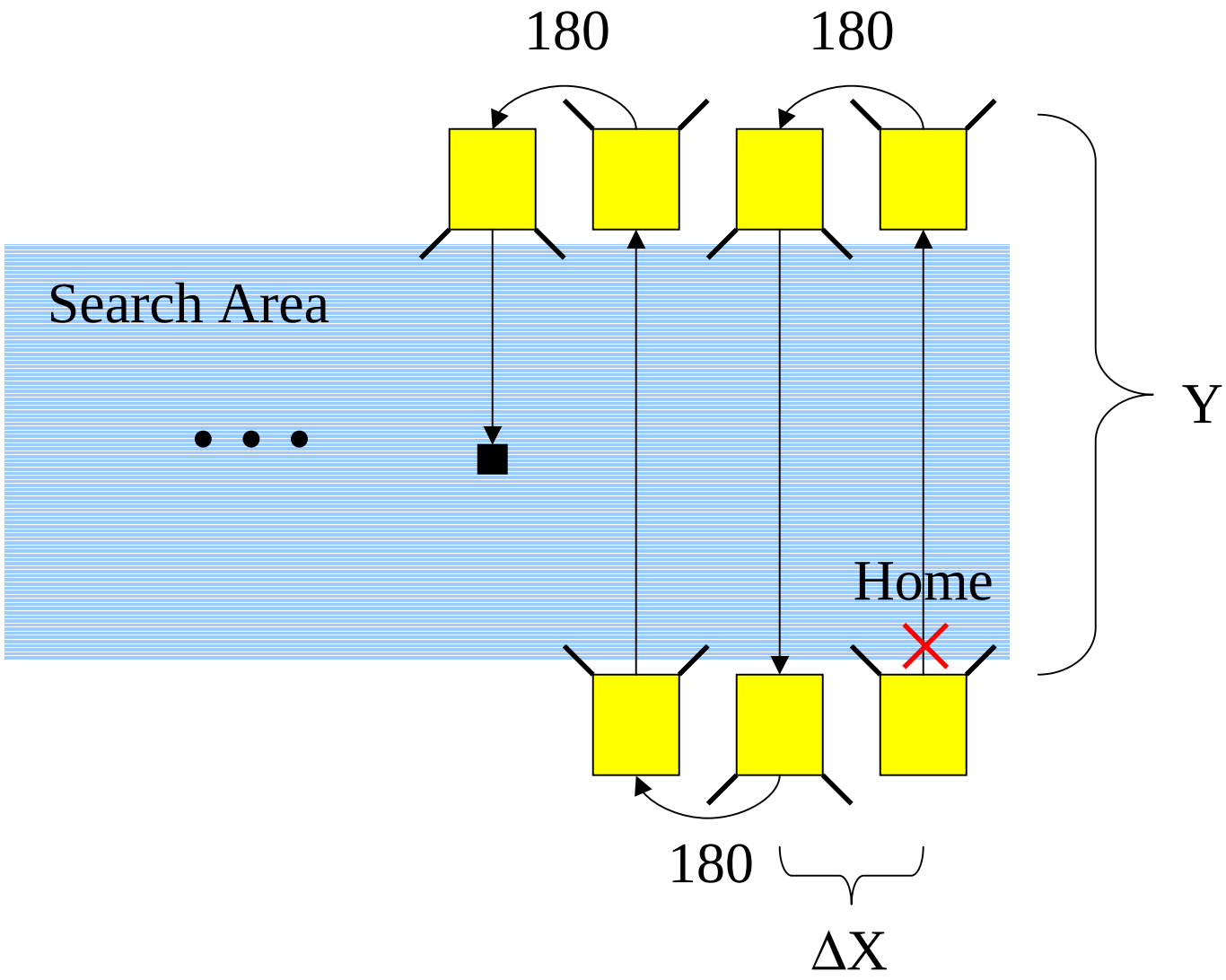
Threads

- Search
  - Run a snake pattern
- Rescue
  - Wait for block, return to starting position/orientation when found.
- Pause
  - Pause/resume Search/Rescue threads
- Display
  - Local and remote real-time display

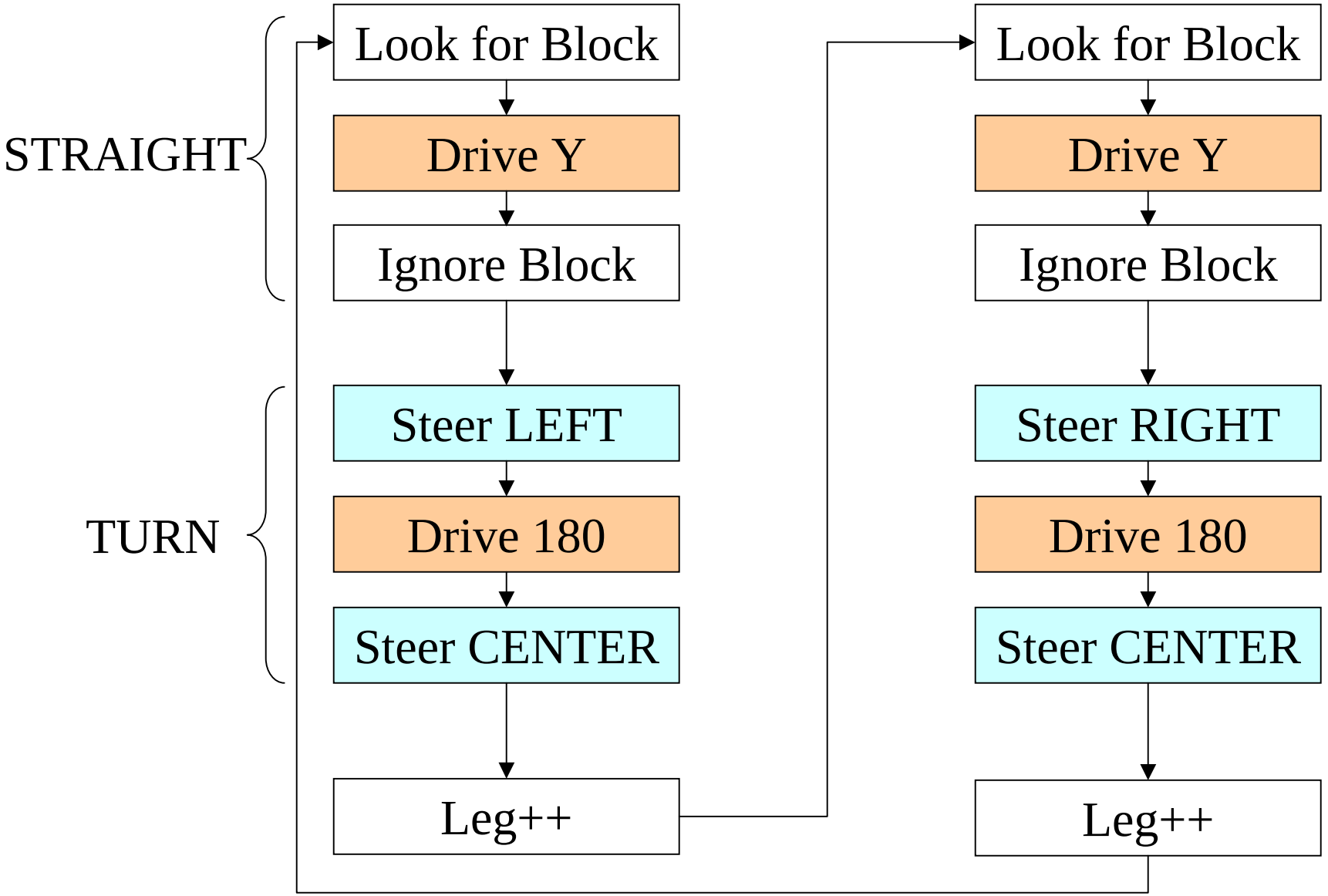
Thread Priorities



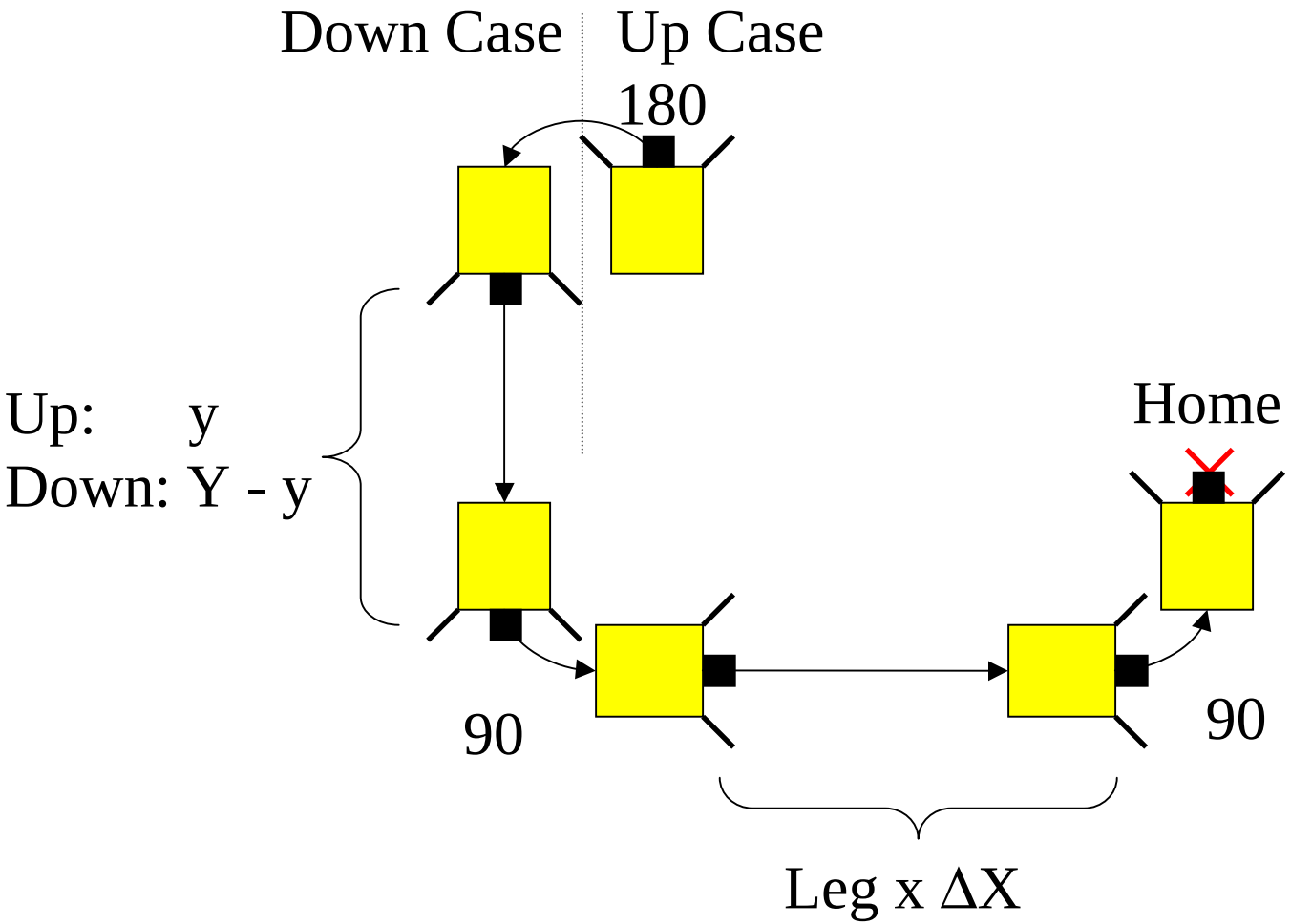
Snake Search Pattern



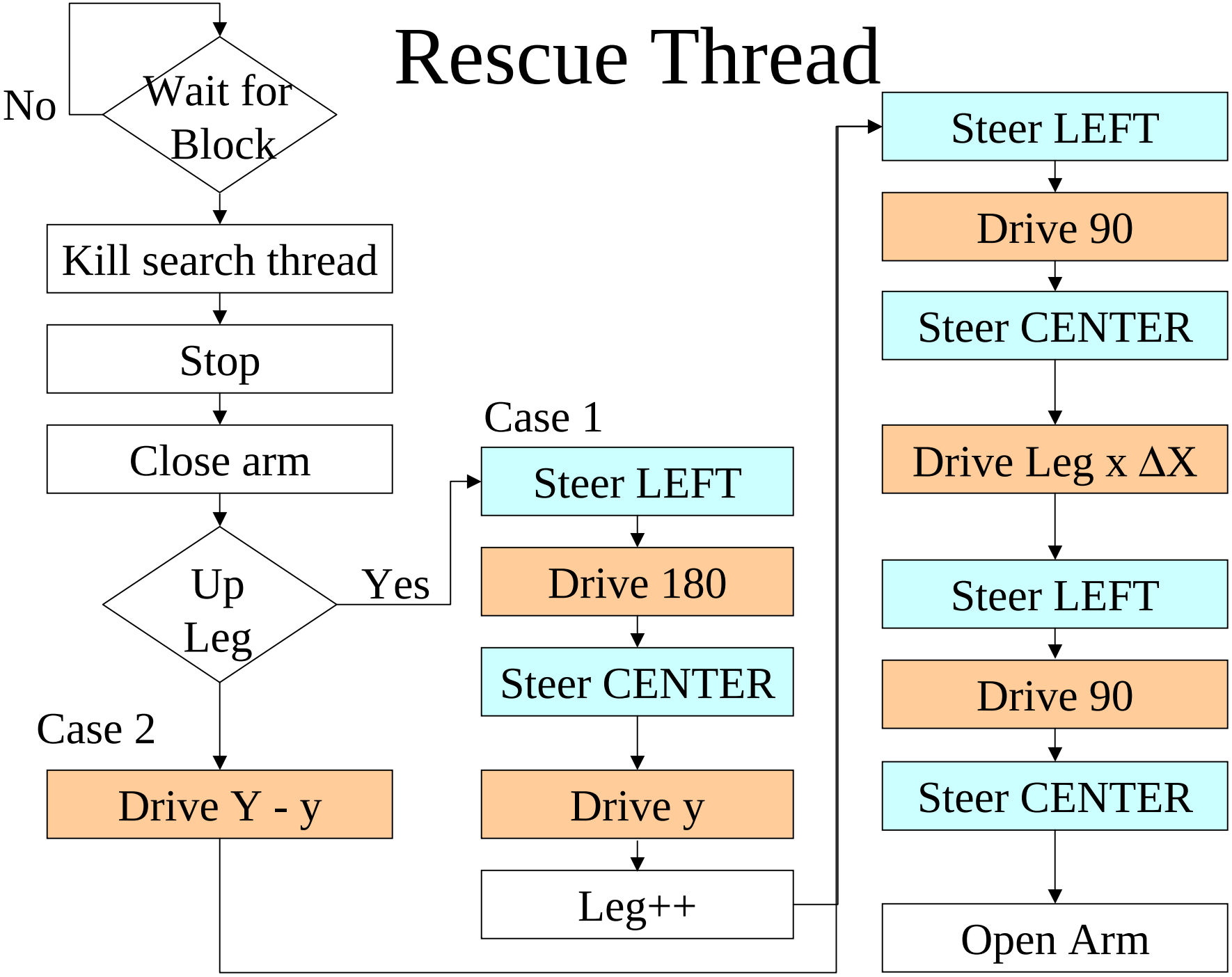
Search Thread



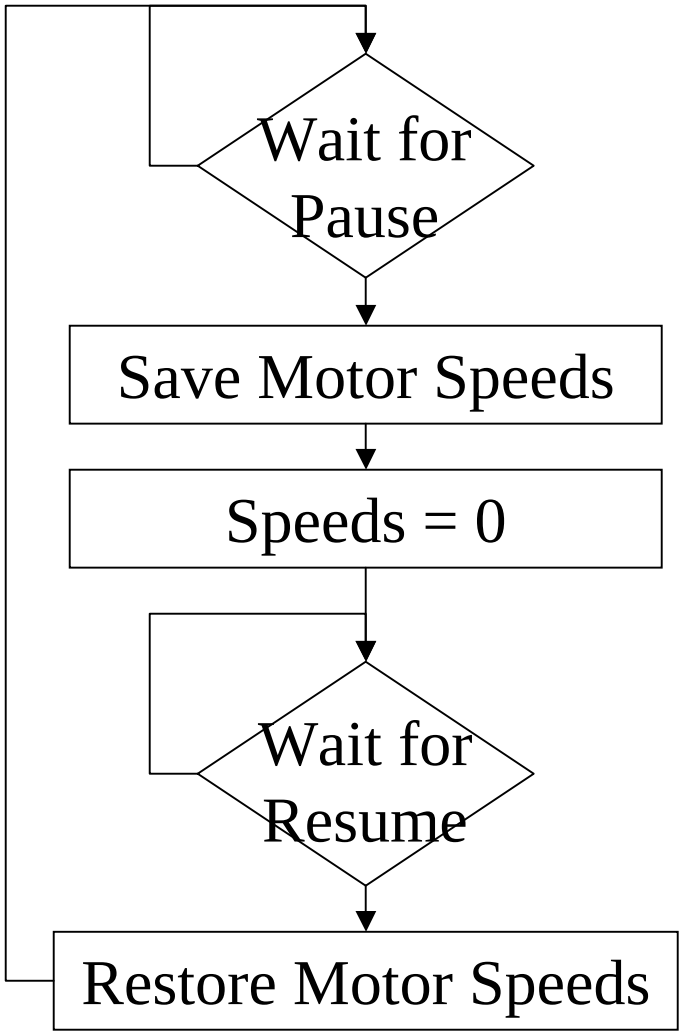
Rescue Pattern



Rescue Thread



Pause Thread

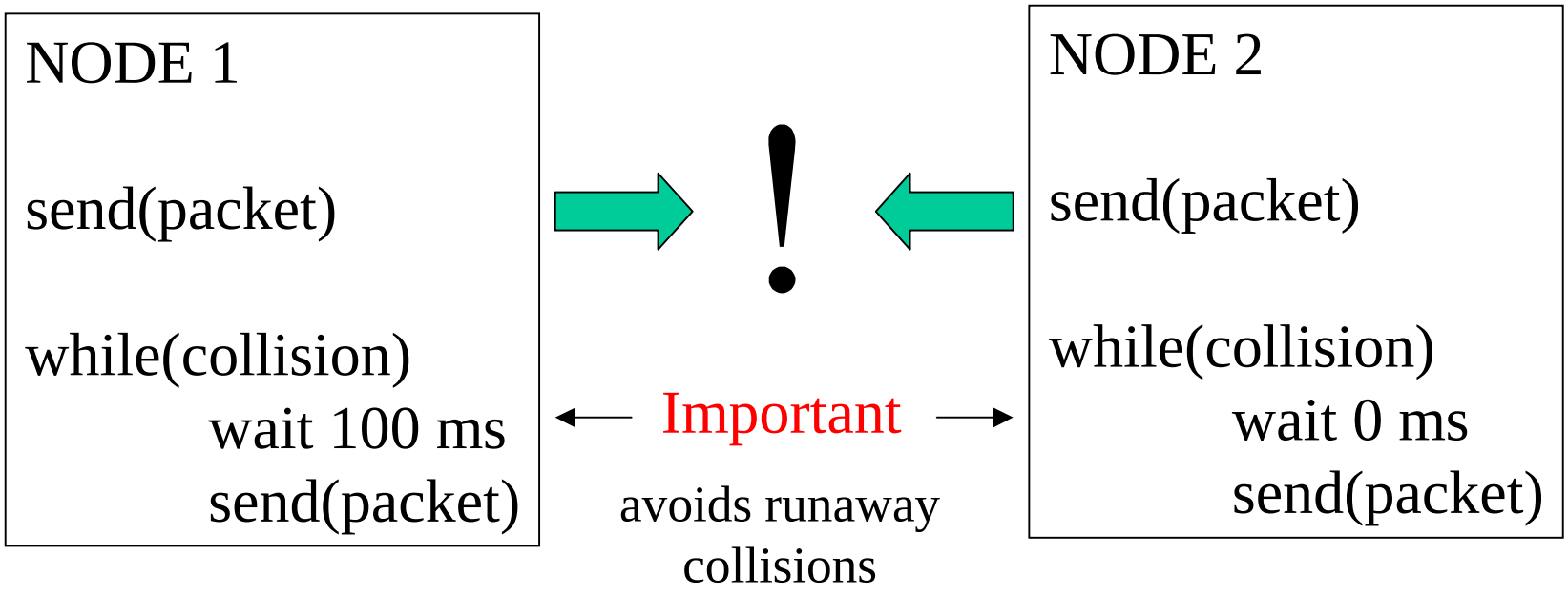


wait_event to let lower priority threads run

Busy wait to block lower priority threads

Collision Handling

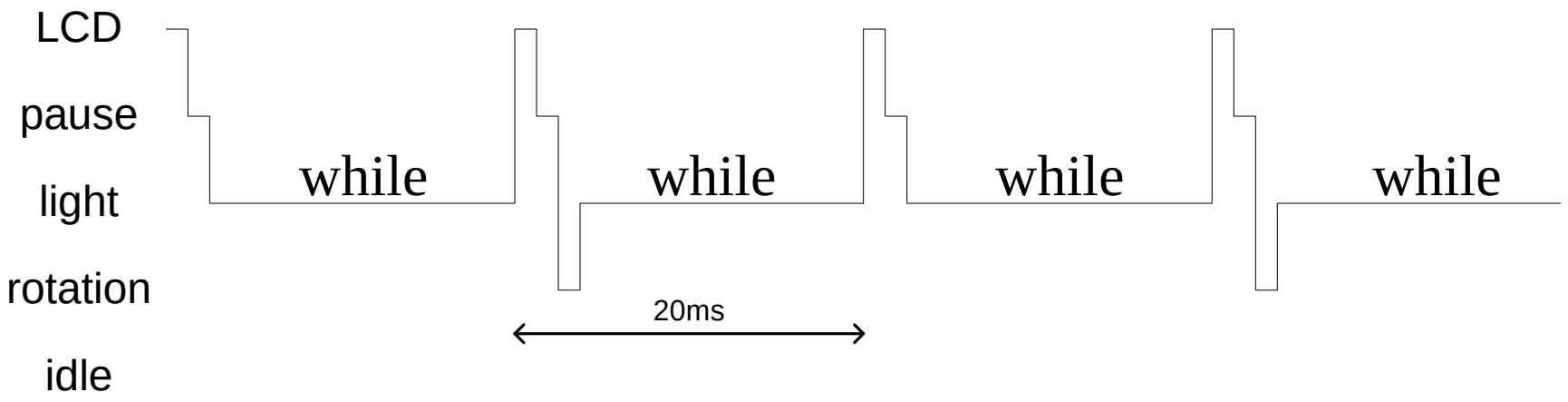
- Pause/Resume important
- Collisions frequent with two-way communication
  - Add retries with “back off”
- Results in high reliability, but not guaranteed

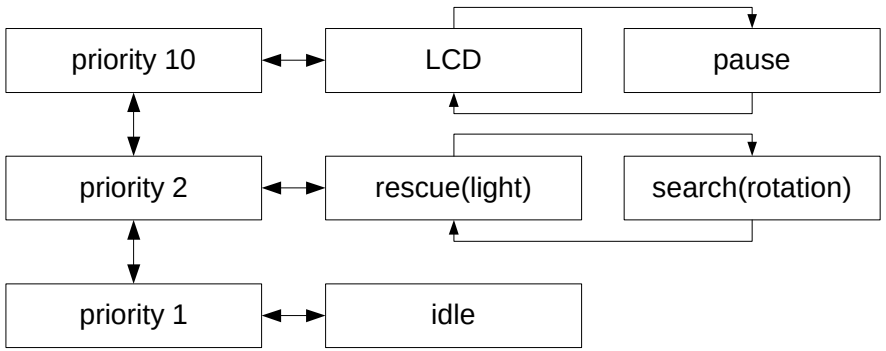


Scheduling

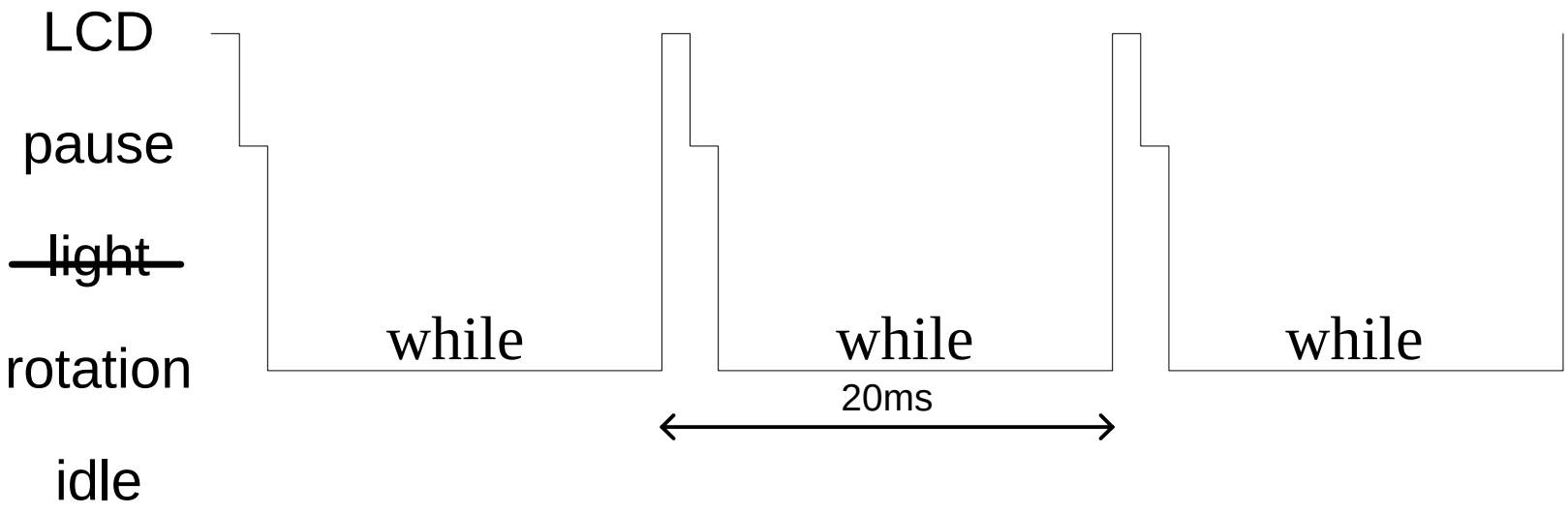
- Wait_event
  - Poll only once every 20ms
  - Allows lower priority threads to run
- While
  - Poll continuously for 20ms
  - Prevents wasting CPU cycles in idle

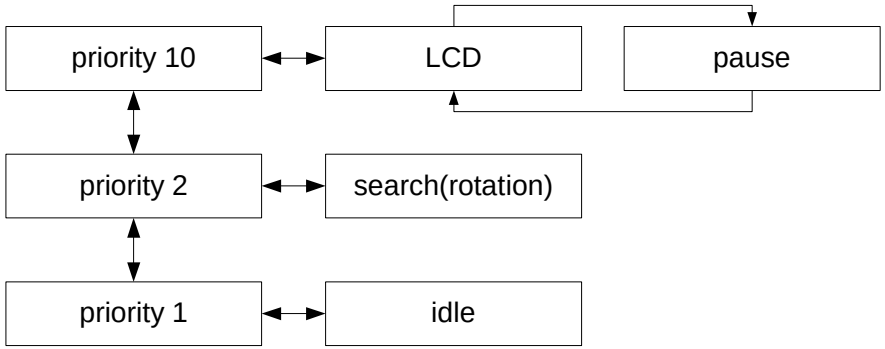
Driving straight



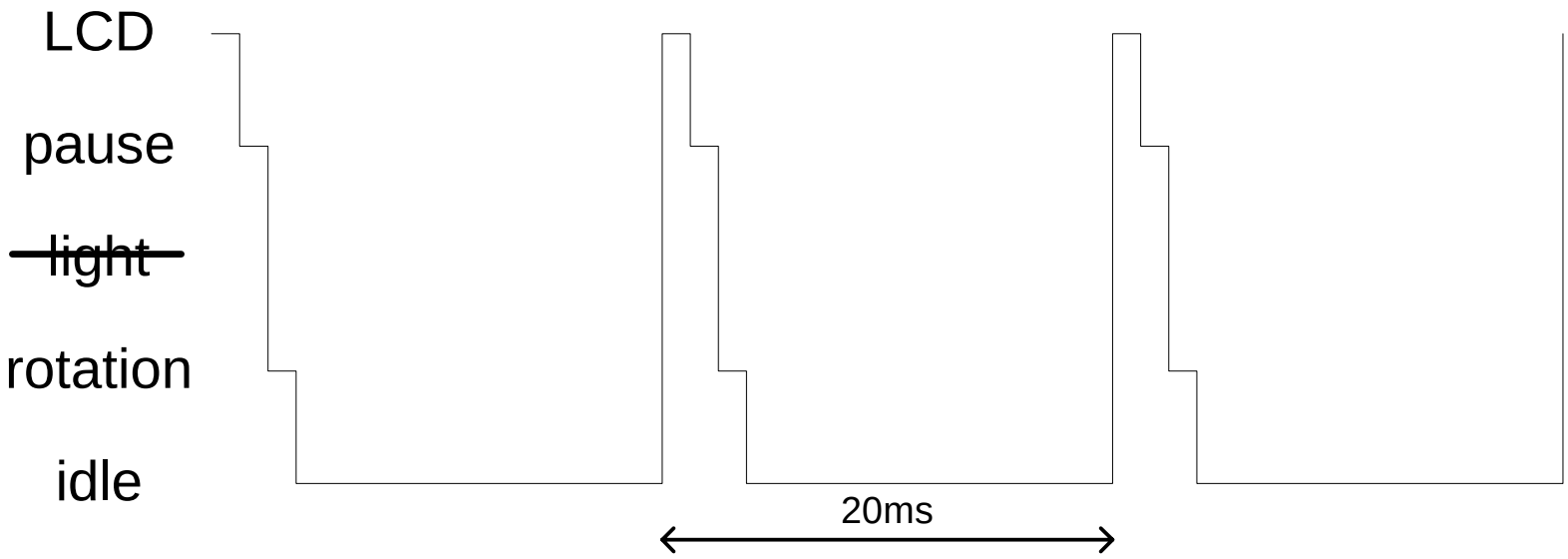


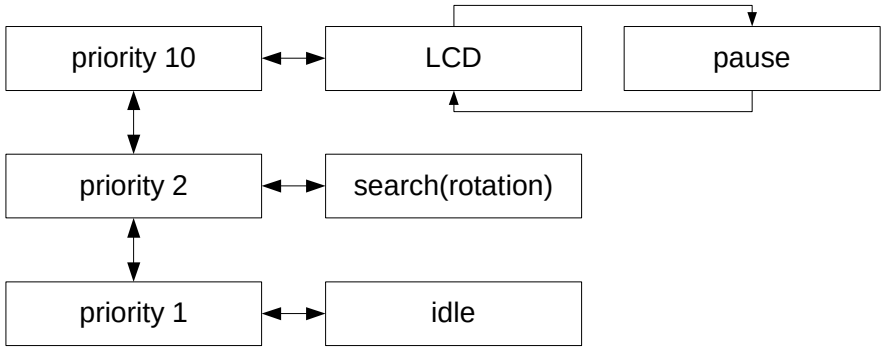
Changing steering





Driving turn





Search and Rescue Robot

Shannon Zelinski
Chris Cortopassi