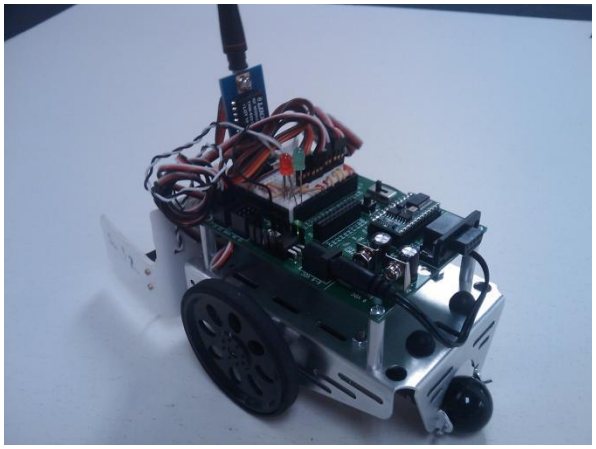
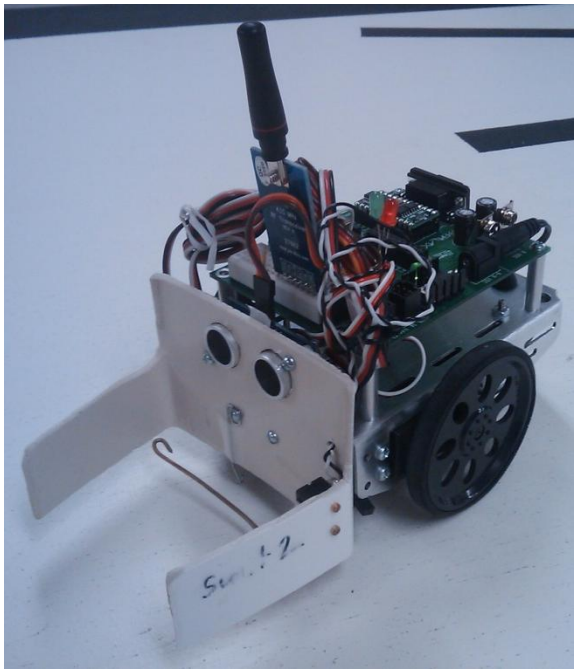
Hardware for Software Engineers Fall 2010

Final Project: LabRat
Design of a Research and Rescue Autonomous Robot



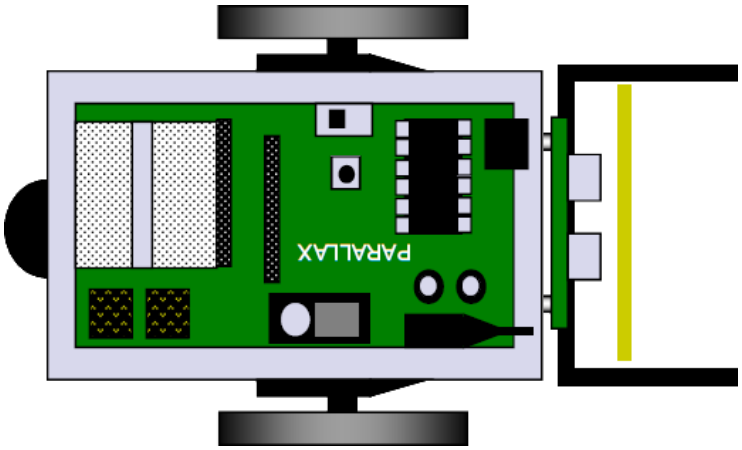
Team LL Ωhmbres
Krishna Ersson
John Peabody

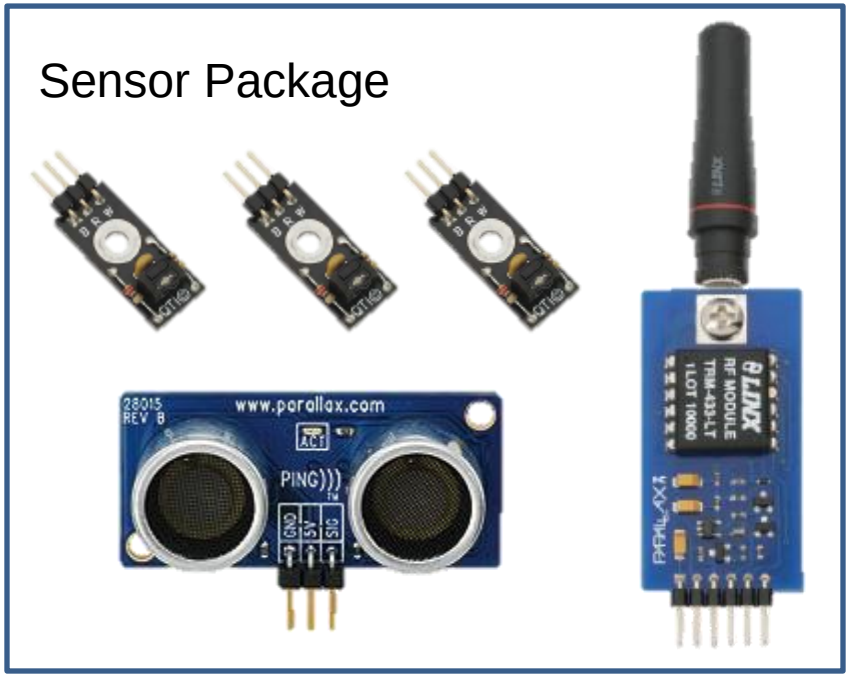
2 December 2010



Outline

- Concept
- Software Design
- System Design
  - Navigation
  - Searching
  - Communications
  - User Interface
- Conclusions



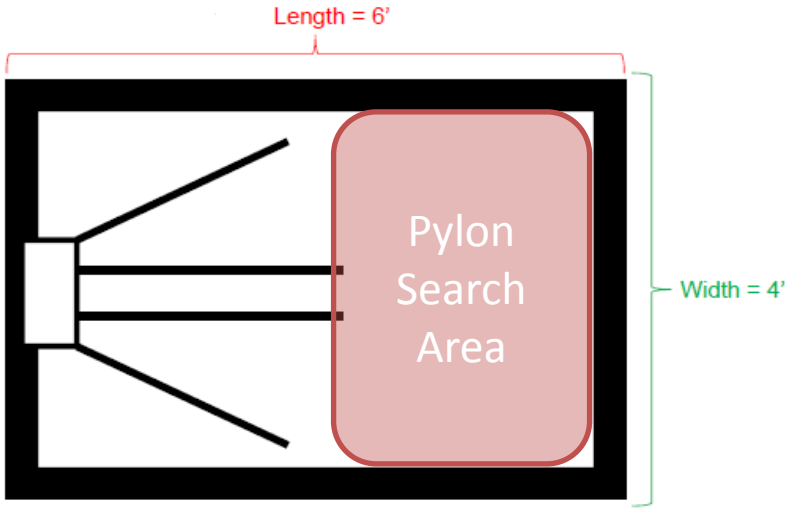


Concept: Search and Rescue

- Locate unknown number of pylons
- Return pylons safely to home area
- Navigate the playing field
  - Avoid edges
- Await further instructions upon completing mission
- Abort and return to home area at any point

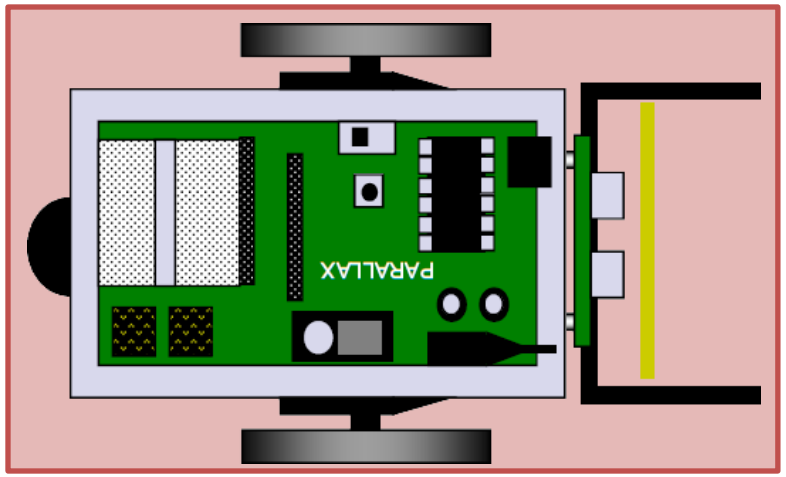
Concept: Assumptions

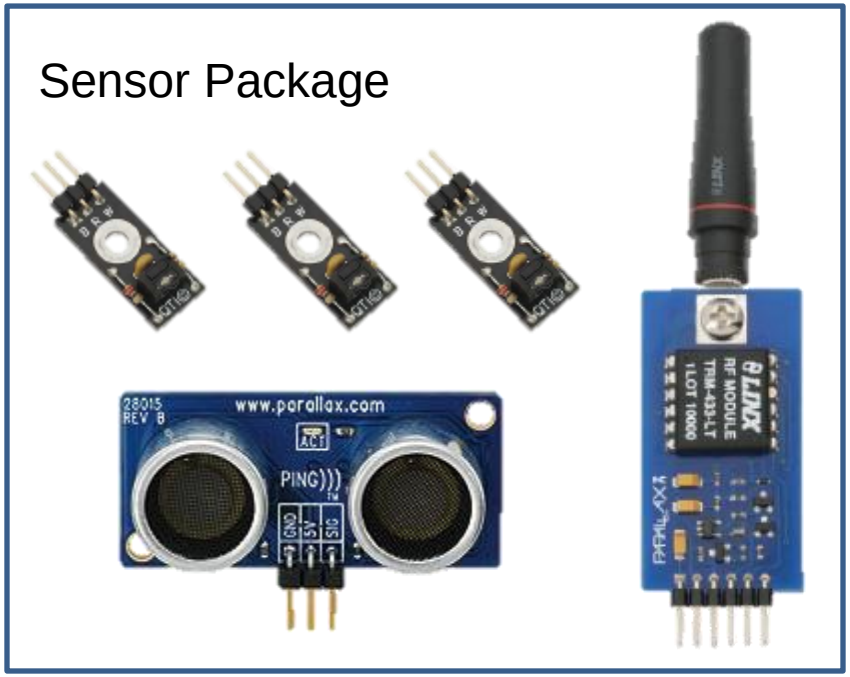
- Number of pylons will be between 1 - 3
- Pylon placement will be in whitespace between end of guidelines and game board edge
- Abort returns the LabRat to home area
- When no pylons are found the LabRat holds position waiting for continue command



Outline

- Concept
- Software Design
- System Design
  - Navigation
  - Searching
  - Communications
  - User Interface
- Conclusions





Software Design

- Decoupled tasks with a **cyclic executive scheduler**
  - Simple and maintainable architecture
  - Clear separation between robot hardware and software subsystems
  - Extensible framework and reusable modules

- Potential drawbacks
  - Not the most efficient (code size)
  - Not as fast as a tightly coupled control loop

Software Design

Code setup, calibration, and initialization global state space

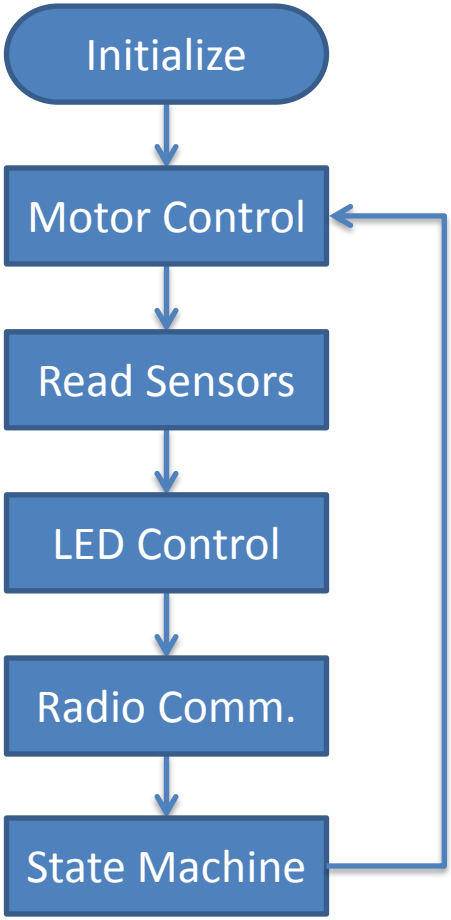
Pulse the servos

Read the sensors; each call reads only one sensor

Turn the LEDs on or off depending on the desired state

Send a ping and check for a command

Main control logic state machine

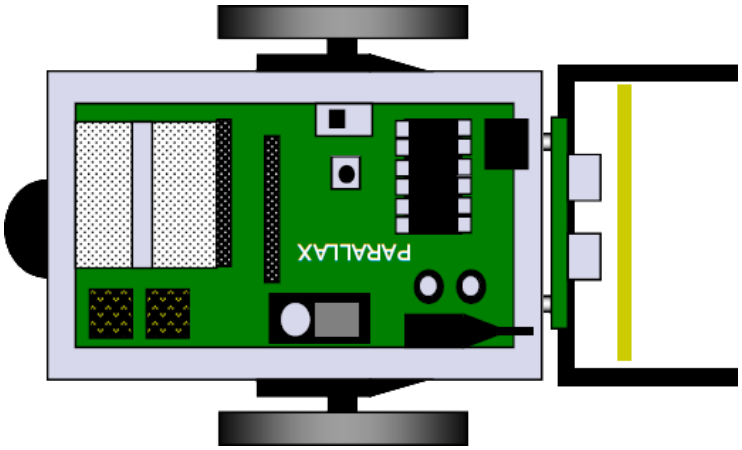


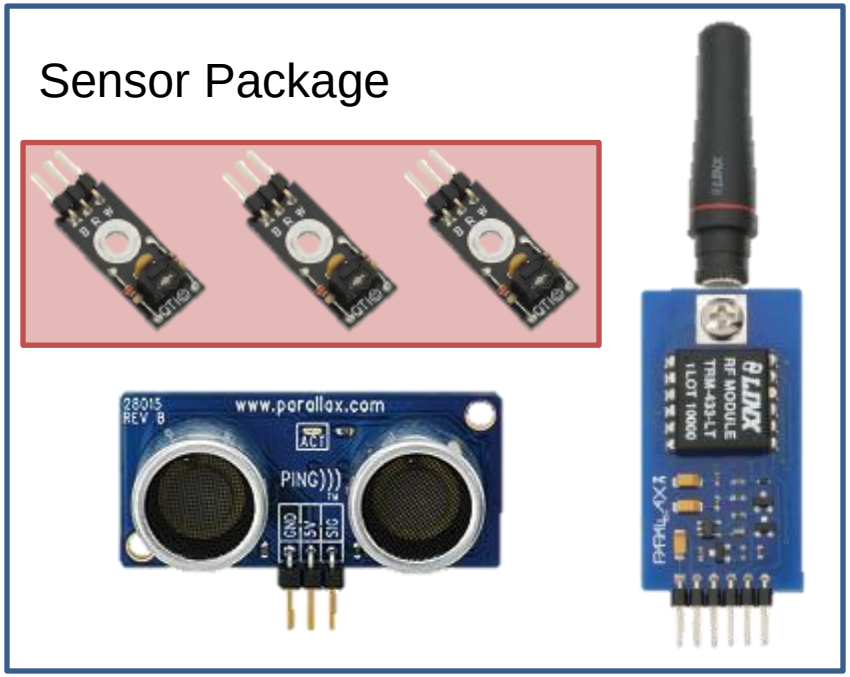
PBASIC implementation uses subroutines for each task:

```
' -----[ Main Program Loop ]-----
Main:
  DO
    GOSUB MotorControl
    GOSUB ReadSensors
    GOSUB LEDControl
    GOSUB RadioComm
    GOSUB StateMachine
  LOOP
  END
```

Outline

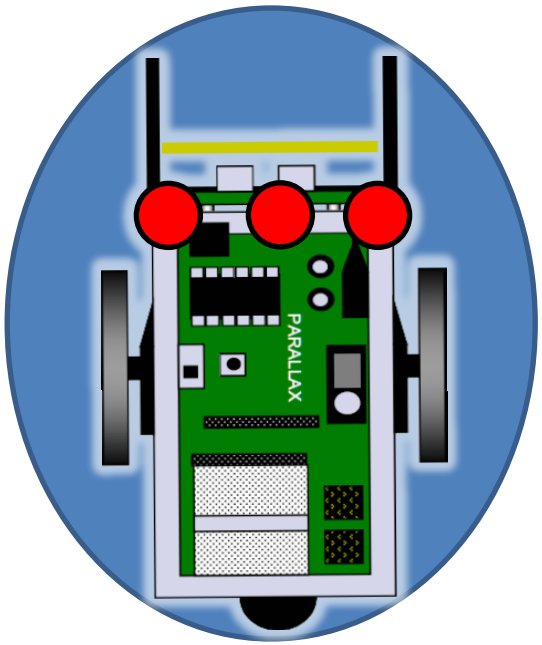
- Concept
- Software Design
- System Design
  - Navigation
  - Searching
  - Communications
  - User Interface
- Conclusions

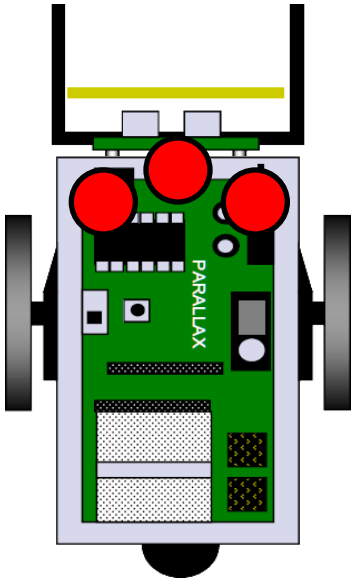


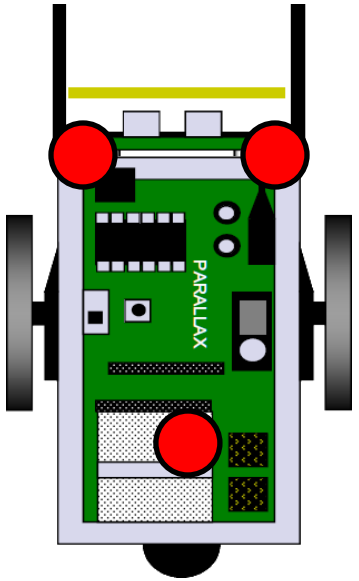


Robot Design: Navigation

- Tradeoff with QTI sensor placement
  - Triangle or line
- Distinguish between guide lines, edge lines, and home

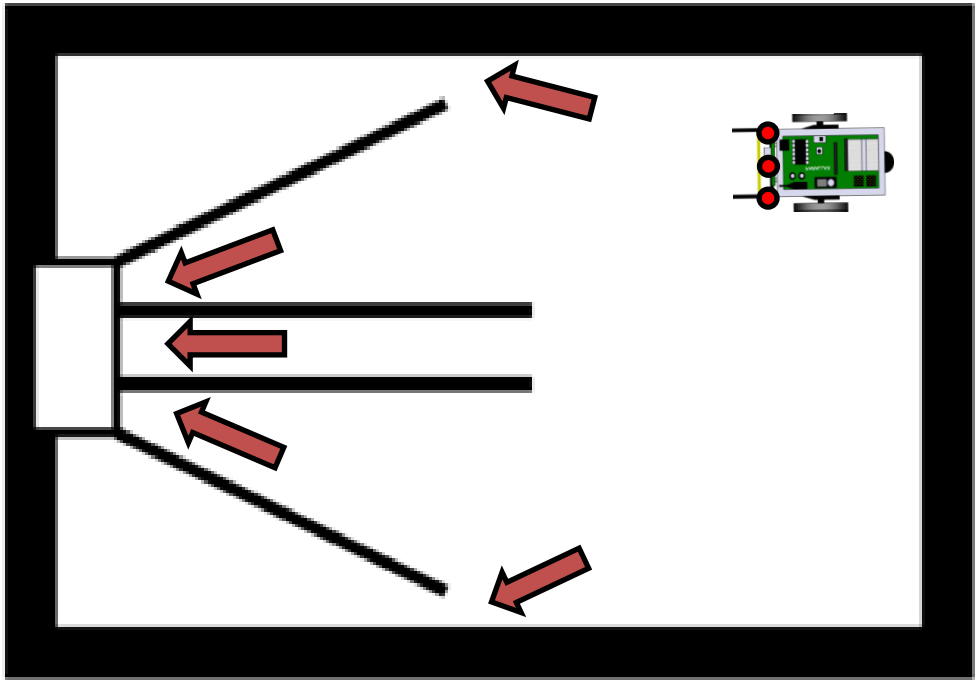




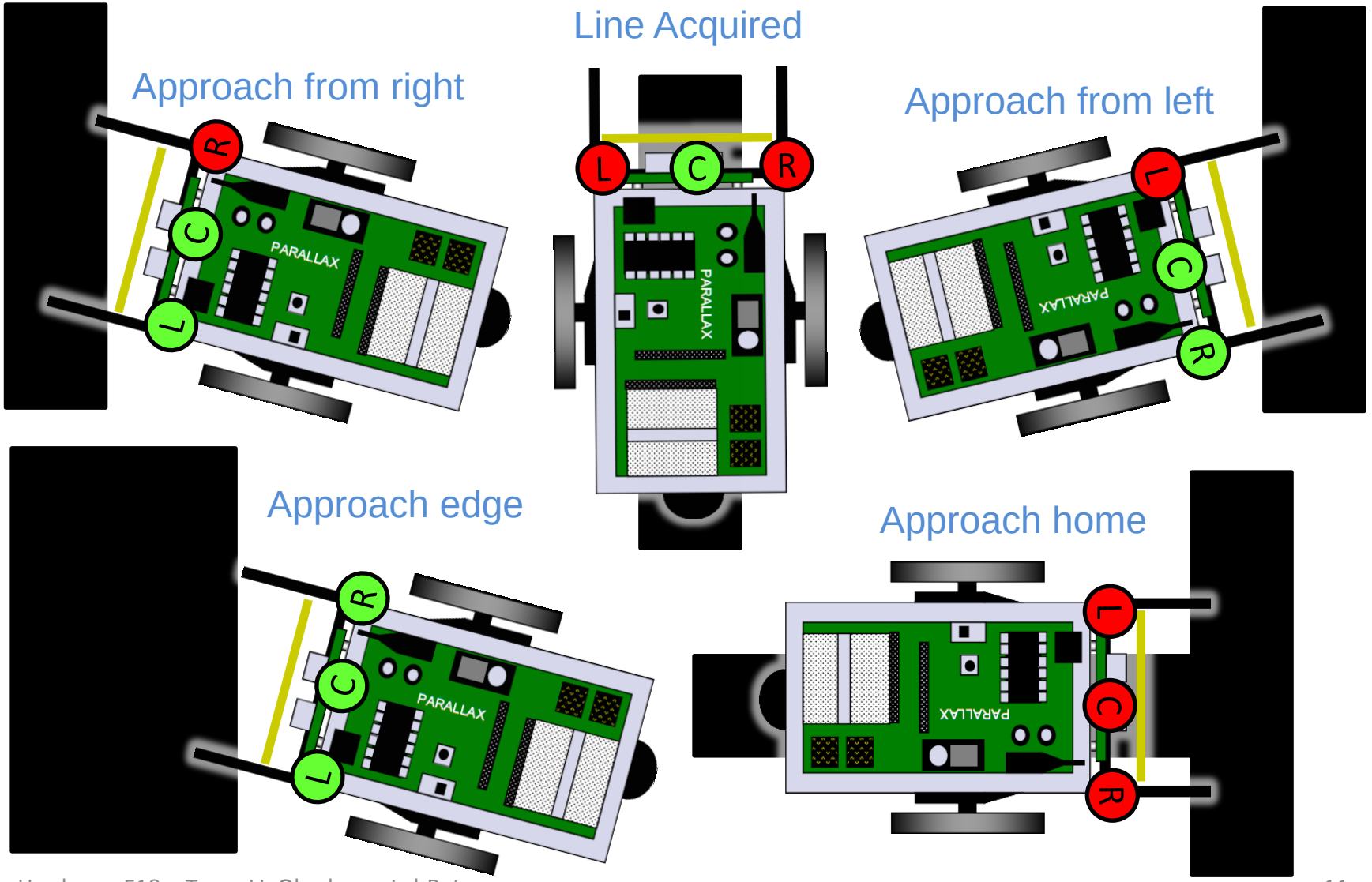


Robot Design: Navigation

- Avoid or mitigate hazardous areas
  - Approaching home straight on
  - Missing guide lines and getting to the side of the home area
- Slightly turning left on returns mitigates these areas



Robot Design: Navigation



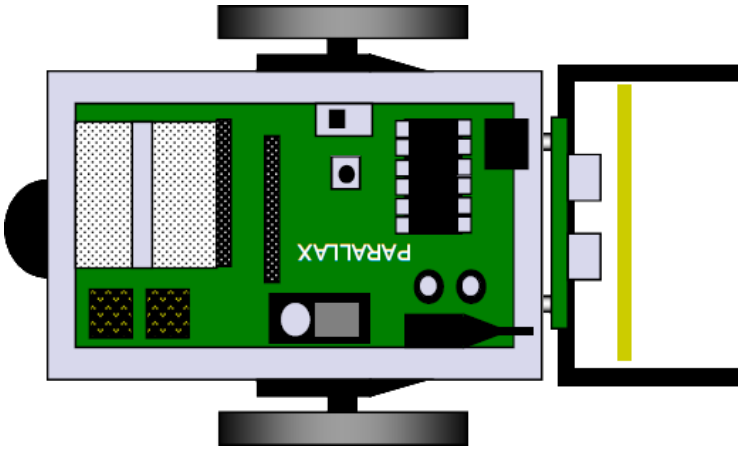
Robot Design: Navigation

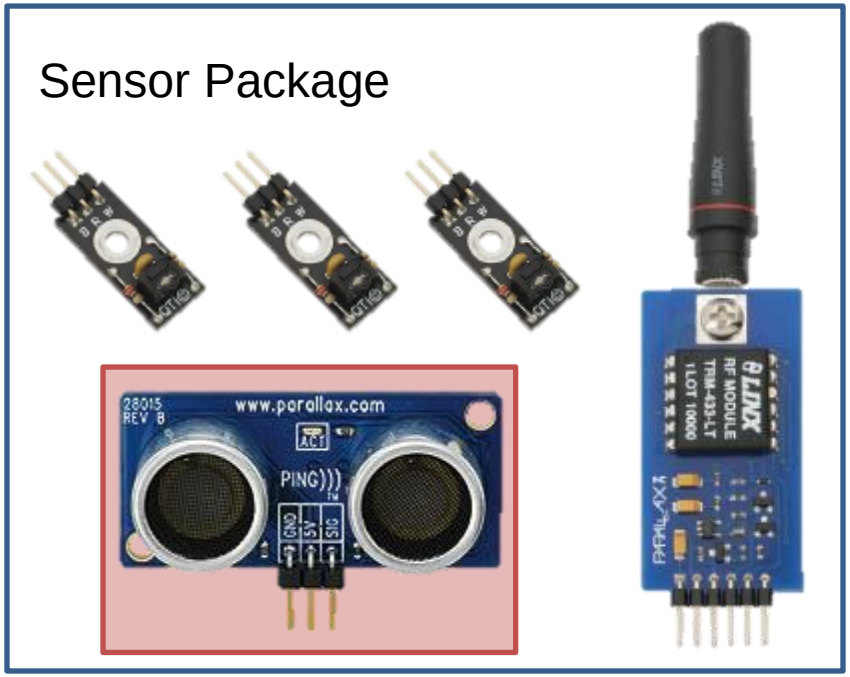
- Direction bit allows reuse of navigation subroutines
  - **LineDir = In** for returning pylons and aborting
  - **LineDir = Out** for returning to pylon search space

- Navigation subroutines also used to ensure robot safety while searching
  - When not tracking a pylon navigation subroutines are used to avoid edges

Outline

- Concept
- Software Design
- System Design
  - Navigation
  - Searching
  - Communications
  - User Interface
- Conclusions



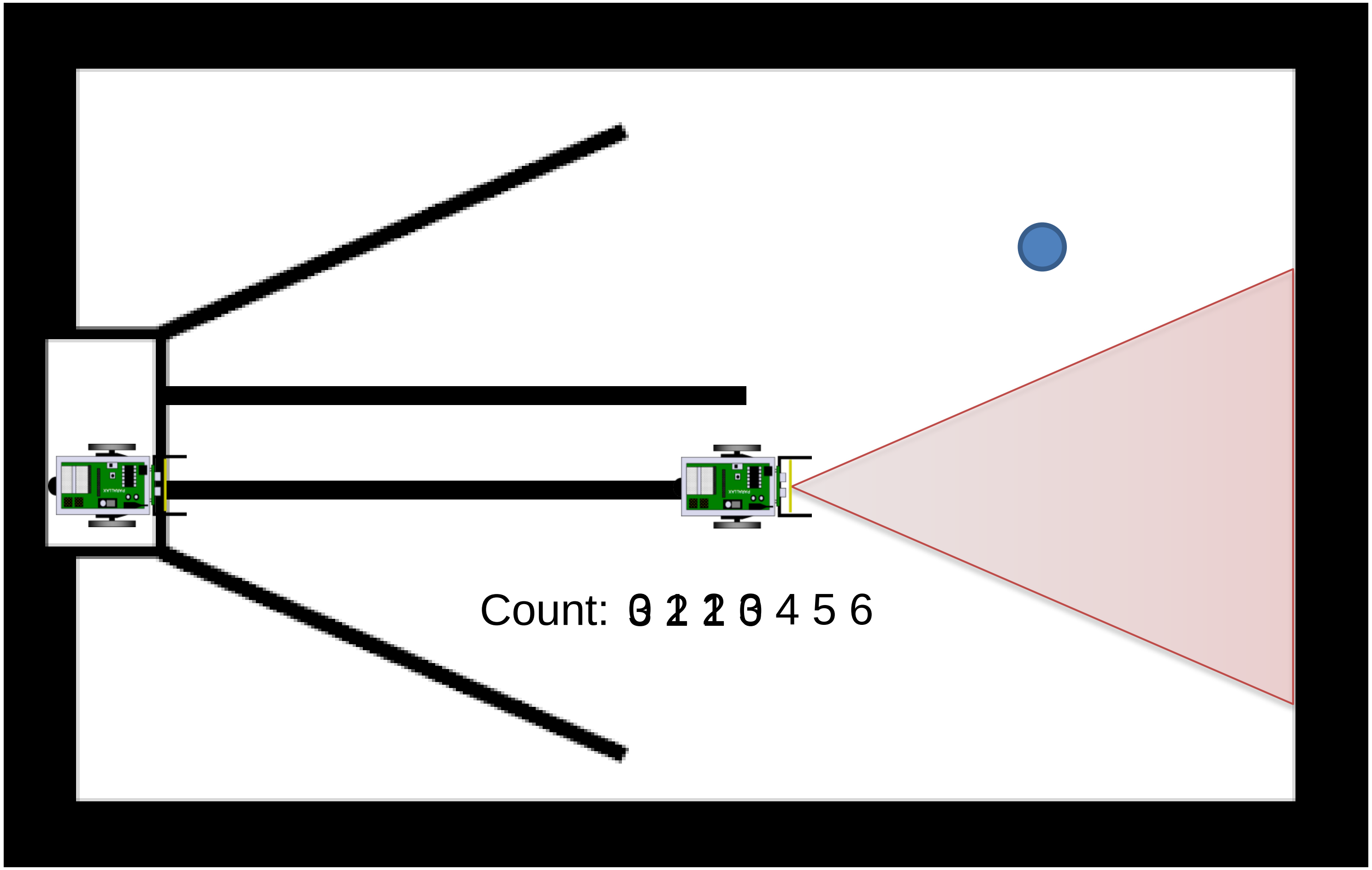


Robot Design: Searching

- Sonar search algorithm
  - Spin CCW until an object is detected closer than 3.5 ft
  - Keep spinning until the object is lost, measuring the amount of time the object remains in view
  - Reverse spin direction and spin back for half the measured time
  - Drive straight until the pylon is docked or we lose track

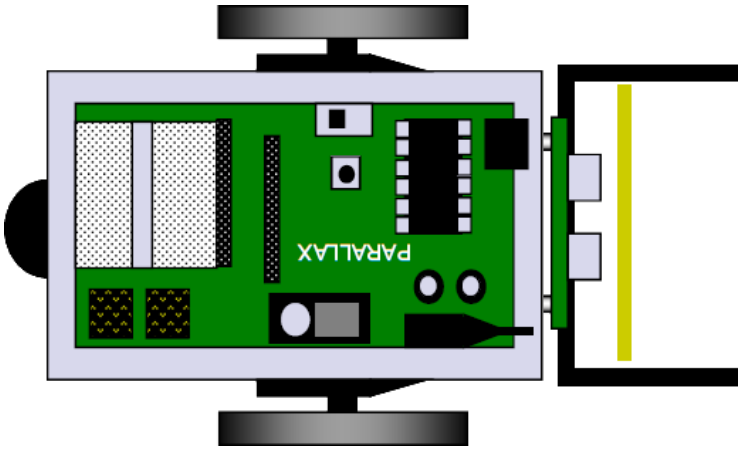
- Relocate once if no pylons found
  - Overcomes sonar cross section deficiency at longer ranges

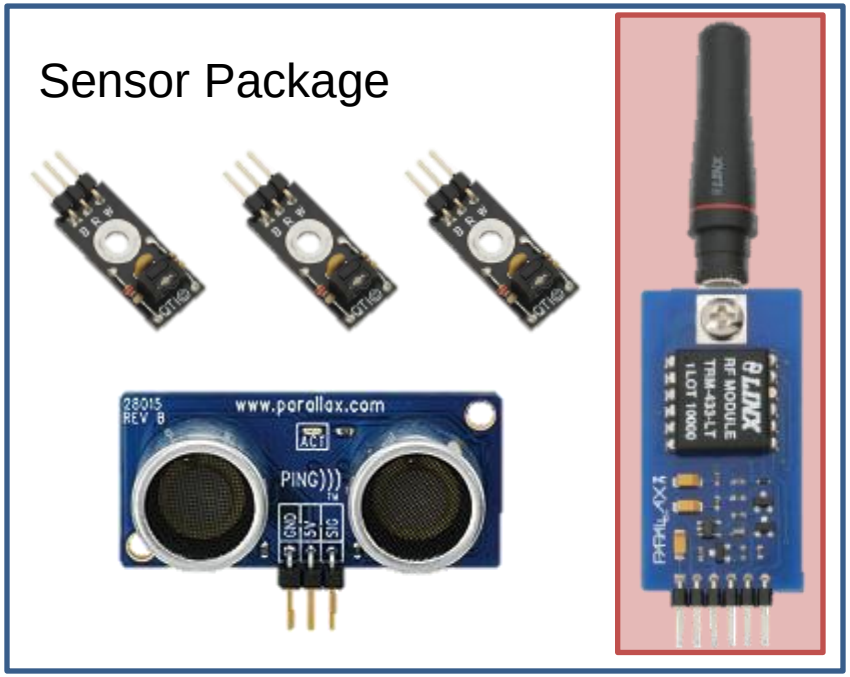
Robot Design: Searching



Outline

- Concept
- Software Design
- System Design
  - Navigation
  - Searching
  - Communications
  - User Interface
- Conclusions





Robot Design: Communications

- Employed minimalist lightweight protocol
  - Three-byte packets
    - 1 sync byte
    - 1 data byte
    - 1 data verification byte
      - Duplicate of the data byte

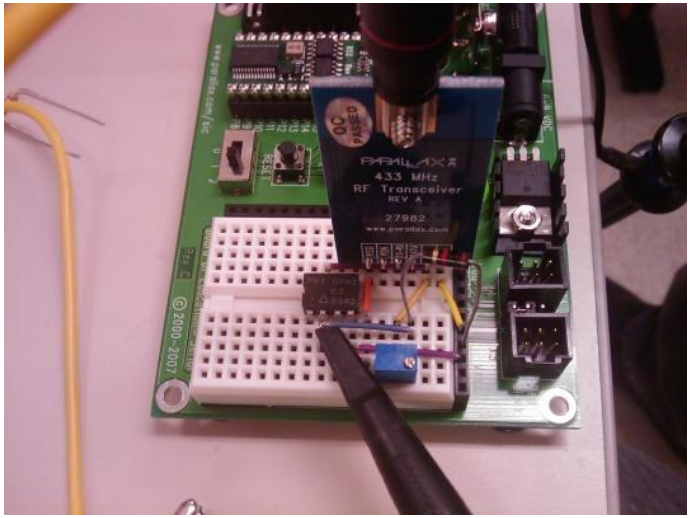
LabRat	Base Station	Java User Interface
Send ping with current state	Receive ping (500 ms)	Receive robot state (1 s)
Receive command (3 ms)	Send command	Send user cmd
Send ack	Receive ack (10 ms)	Receive command OK
	Send command OK to UI	
	Send robot state to UI	
	Receive UI cmd (50 ms)	

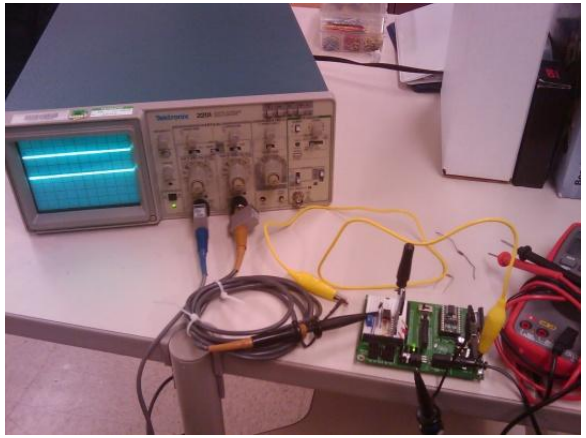
Robot Design: Communications

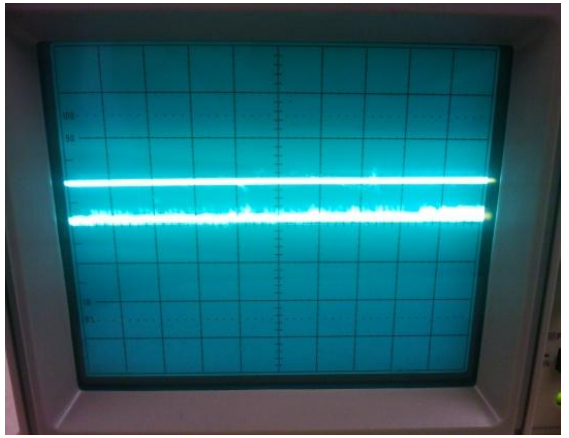
- SERIN command does not timeout due to radio noise
  - Noise generates random serial data
  - Short timeouts (less then ~5 ms) avoid issue
  - Long timeouts allow for fault tolerant system operation

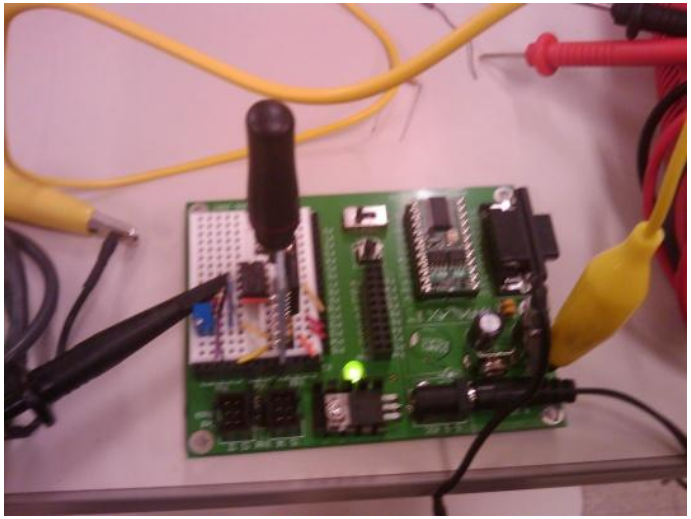
LabRat	Base Station	Java User Interface
Send ping with current state	Receive ping (500 ms)	Receive robot state (1 s)
Receive command (3 ms)	Send command	Send user cmd
Send ack	Receive ack (10 ms)	Receive command OK
	Send command OK to UI	
	Send robot state to UI	
	Receive UI cmd (50 ms)	

Robot Design: Communications

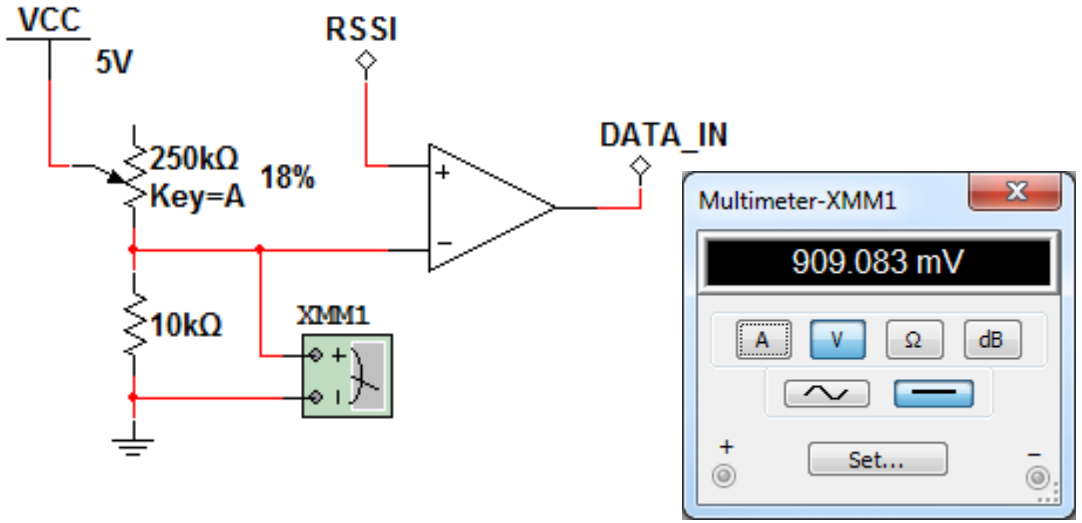




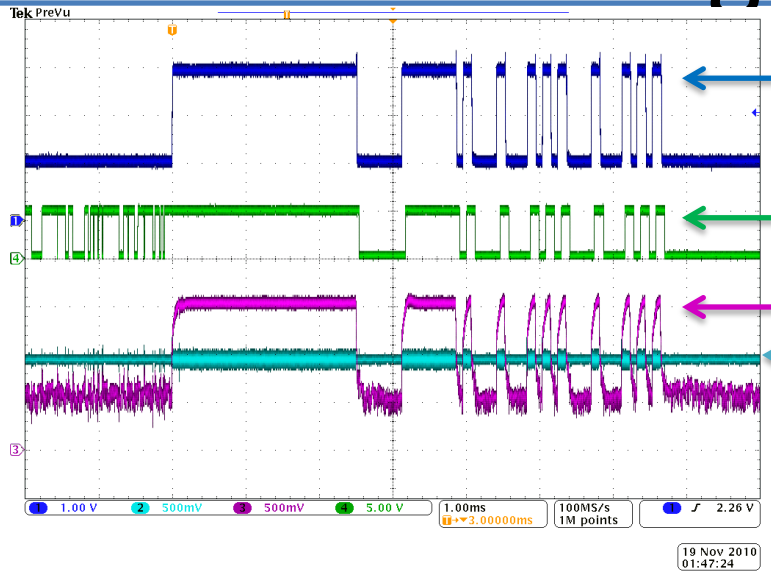


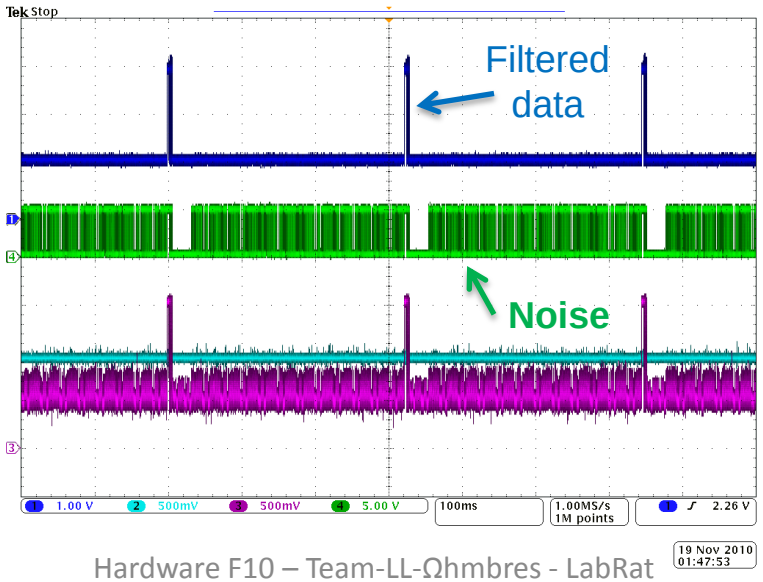


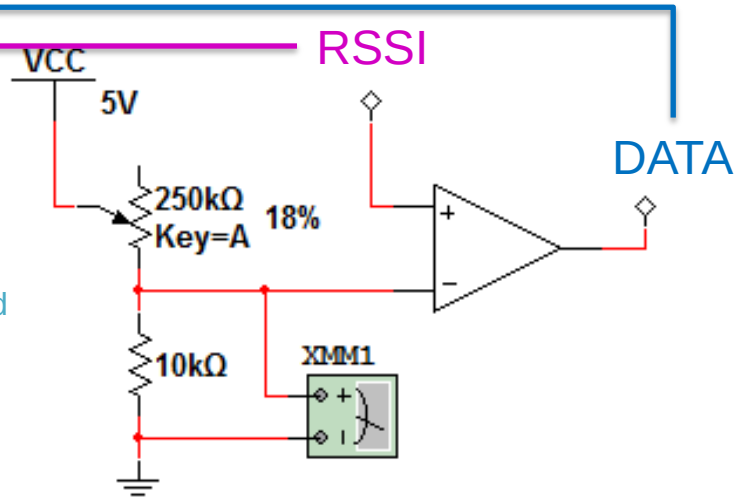
Noise Filter Circuit



Robot Design: Communications



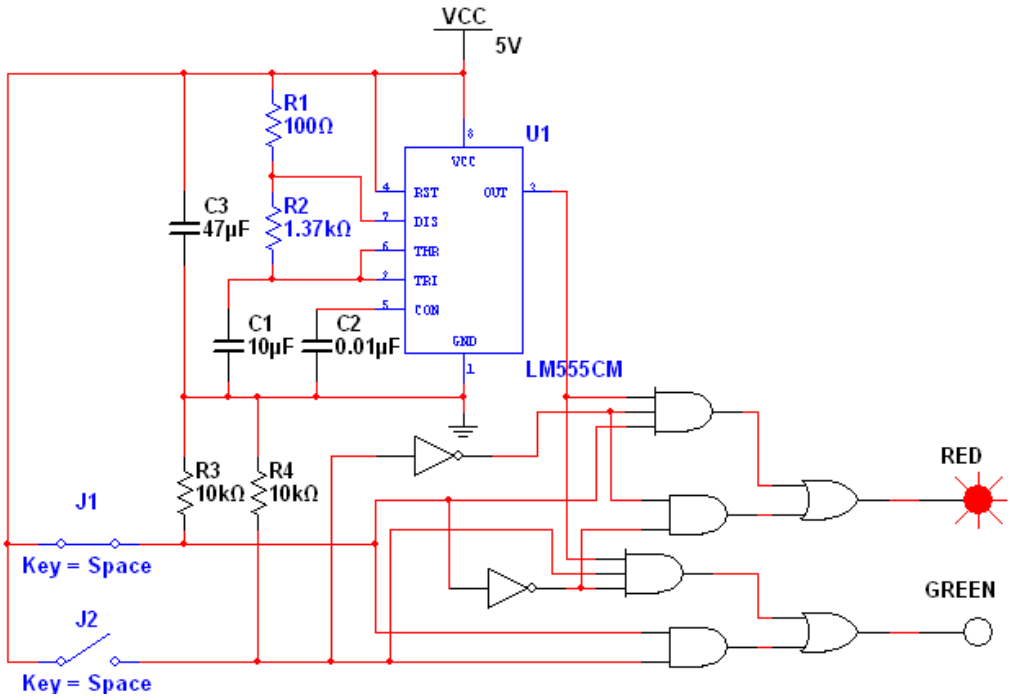




- Zoomed-out view shows noisy data line
  - The RSSI pin voltage goes above $V_{Threshold}$ when real data is present
  - The DATA output from the comparator is fed to the Stamp processor
  - Without the noise, the timeout feature of the SERIN command works correctly

Robot Design: Communications

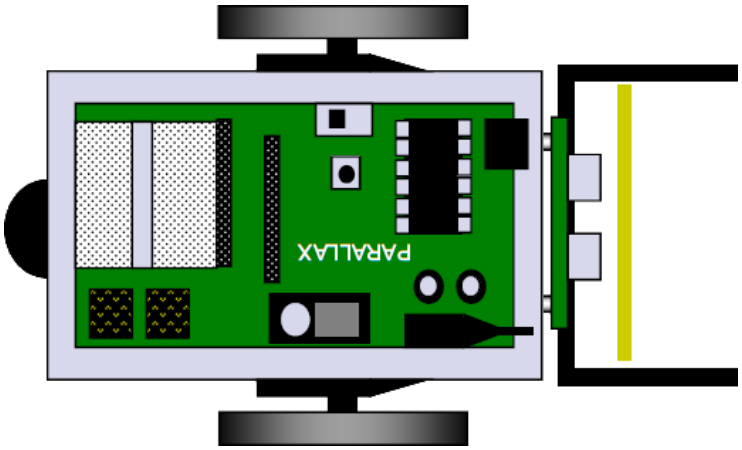
- Hardware vs. Software tradeoff
  - 2 Bit control for both
  - Set it and forget it or continuous updating

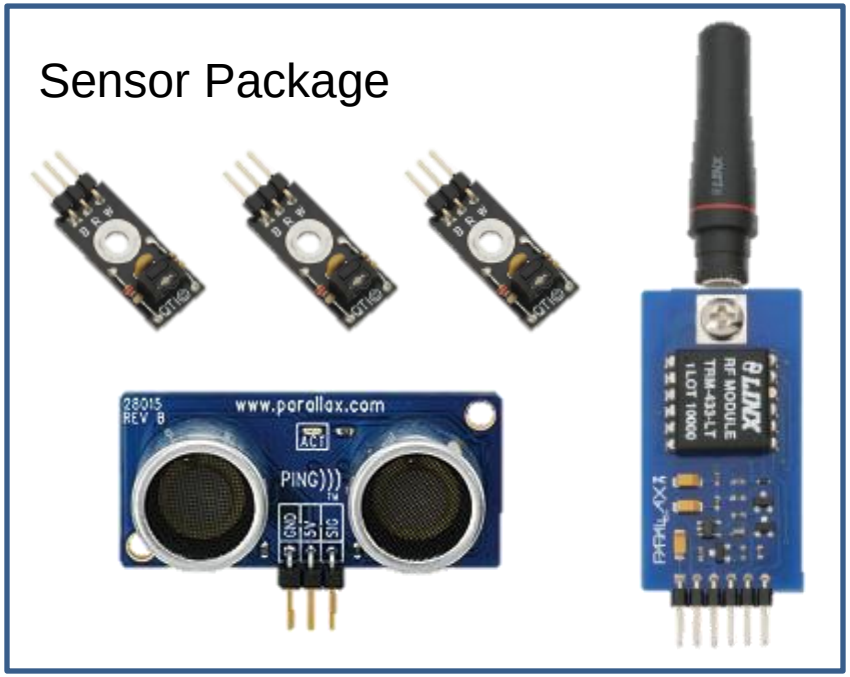



```
' -----[ LED Control ]----------------------
LEDControl:
  SELECT ledRedState
    CASE LedOff: LOW LED_R      'if LED is set
    CASE LedOn:  HIGH LED_R     'if LED is set
    CASE LedBlink:              'if LED is set
      IF ledStateCounter < LedBlinkRate THEN
        LOW LED_R
      ELSE
        HIGH LED_R
      ENDIF
  ENDSELECT

  SELECT ledGreenState
    CASE LedOff: LOW LED_G      'if LED is set
    CASE LedOn:  HIGH LED_G     'if LED is set
    CASE LedBlink:              'if LED is set
      IF ledStateCounter < LedBlinkRate THEN
        LOW LED_G
      ELSE
        HIGH LED_G
      ENDIF
  ENDSELECT

  ledStateCounter = ledStateCounter + 1
  IF ledStateCounter >= LedBlinkRate * 2 THEN
    ledStateCounter = 0
  ENDIF
  RETURN
```

Outline

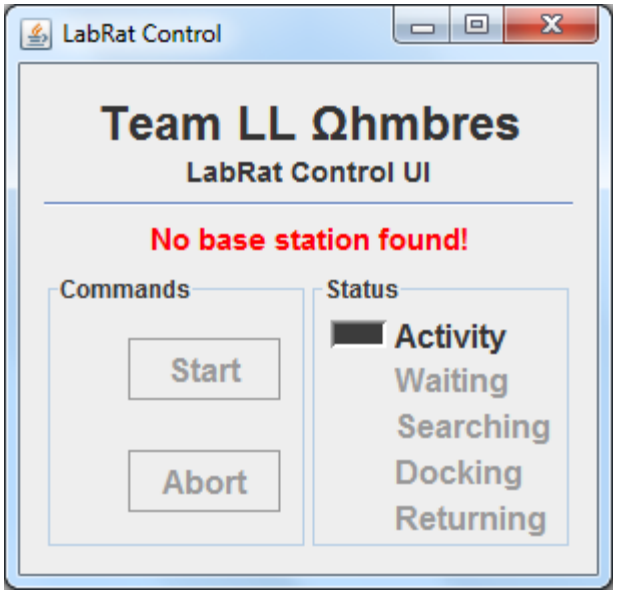
- Concept
- Software Design
- System Design
  - Navigation
  - Searching
  - Communications
  - User Interface
- Conclusions



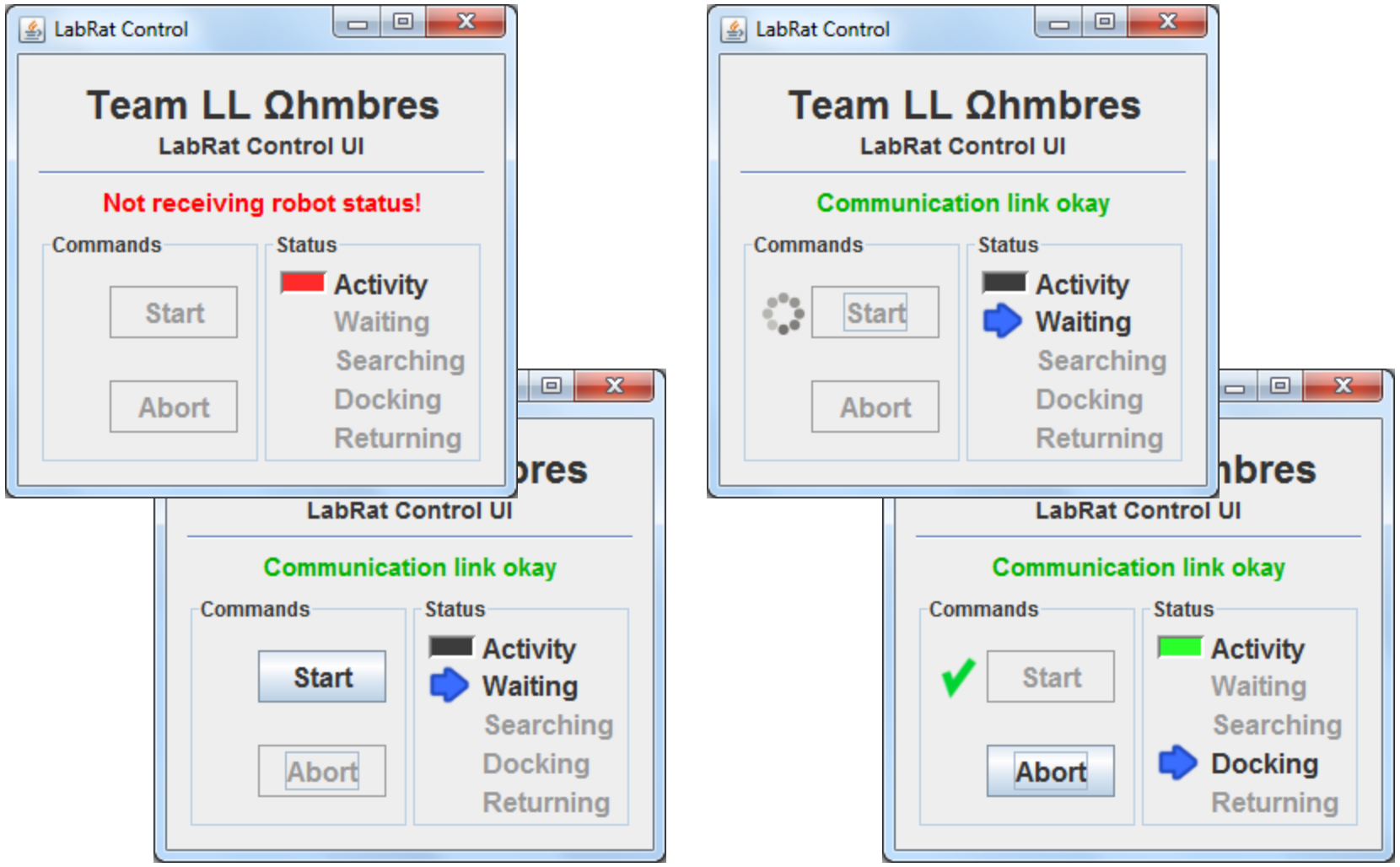


Robot Design: User Interface

- Sending commands
  - Compact interface for selecting command to send
  - Command acknowledgement
- Displaying LabRat status
  - Radio activity indicator
  - State indicator
- UI design driven by the comm. architecture
  - SERIN timeouts allow the UI and robot to operate independently with a minimal amount of data



Robot Design: User Interface



Conclusions

- EEPROM usage: 100%
- RAM usage: 90%
- CPU limits are challenging
  - No interrupts
  - No precision timing methods, task scheduling HARD
  - No floating point
  - No signed data types (negative numbers in IF)
- System integration exposes unforeseen issues
- Understand component capabilities and constraints
  - Avoids problems and may provide solutions