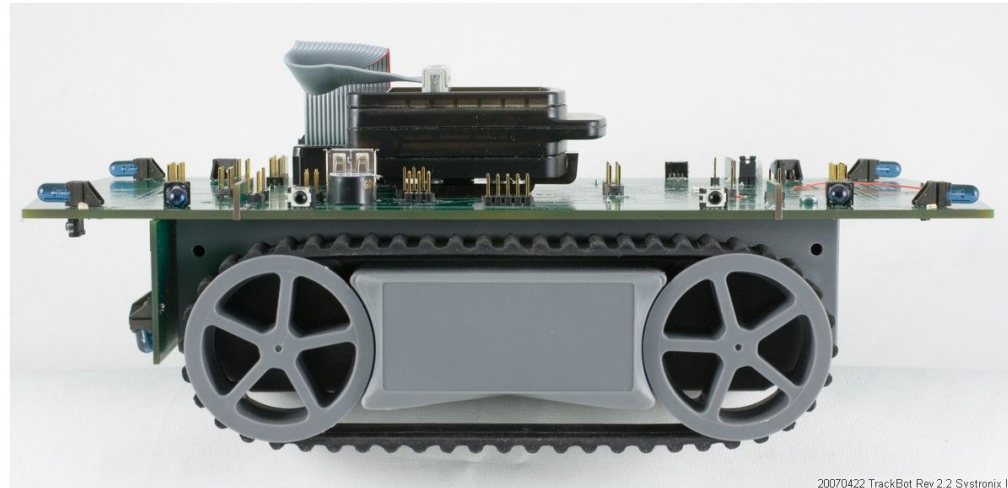


# TrackBot: Designing a Java-powered platform for swarm research and collaboration with wireless sensors



Bruce Boyes  
Systronix Inc  
[www.trackbot.systronix.com](http://www.trackbot.systronix.com)

**SYSTRONIX**  
Embedded Java Spoken Here

# TrackBot Architecture & Design

Based on experience teaching robotics and embedded systems at the University of Utah

TrackBot is designed specifically for university-level education and swarm research.

*TrackBot is not just another small robot!*

# What we'll cover

**TrackBot Cause and Market  
Design Goals**

**Unique Architecture & Sensor Array**

**A little bit about Java**

**Technical Challenges**

**Demo / Video(s) – more Fri @ 2PM**

# What Started all This

3 years teaching CS/CE 4710 at the University of Utah

- Efficiency of Java vs C & Assy
- Robotics with Legos and JCX
- Autonomous heterogeneous robots
- Interest in swarms, collaboration and emergent behaviors
- Robotics standards efforts at OMG

# TrackBot Purpose and Market

Really, nothing like it exists now

- University-level education and research
- Swarms, collaboration, emergent behavior
  - Affordable enough to replace simulation
  - Easy enough to program to be practical
  - OTA deployment and maintenance
- Mobile robots and wireless sensors
  - Huge potential market – 3 D's, security, etc

# Initial Design Goals

- Reasonably priced chassis, but still
  - Able to drive over power cords (many can't!)
  - Usable and affordable built-on sensors
  - User-expandable sensors (no silly limits)
- 802.15.4 radio
- Good battery life
- Support for swarms and collaboration
- Options such as vision, speech, docking

# Initial Design Goals

- Support for a wide variety of “brains”
  - Basic “nervous system” is built in to robot
  - Application brain plugs on to platform
  - No vendor or code lock-in on brain
  - Simple serial interface
  - No realtime requirement on brain, all such is handled by the built-in nervous system

# It's all about swarms

Designed for (affordable) swarm research

- Optimized for research into autonomous agents creating complex systems through decentralized control algorithms
- Works with any host with asynch serial
  - Real-time transducer management is done by the robot, no burden on host
  - host can concentrate on high level applications

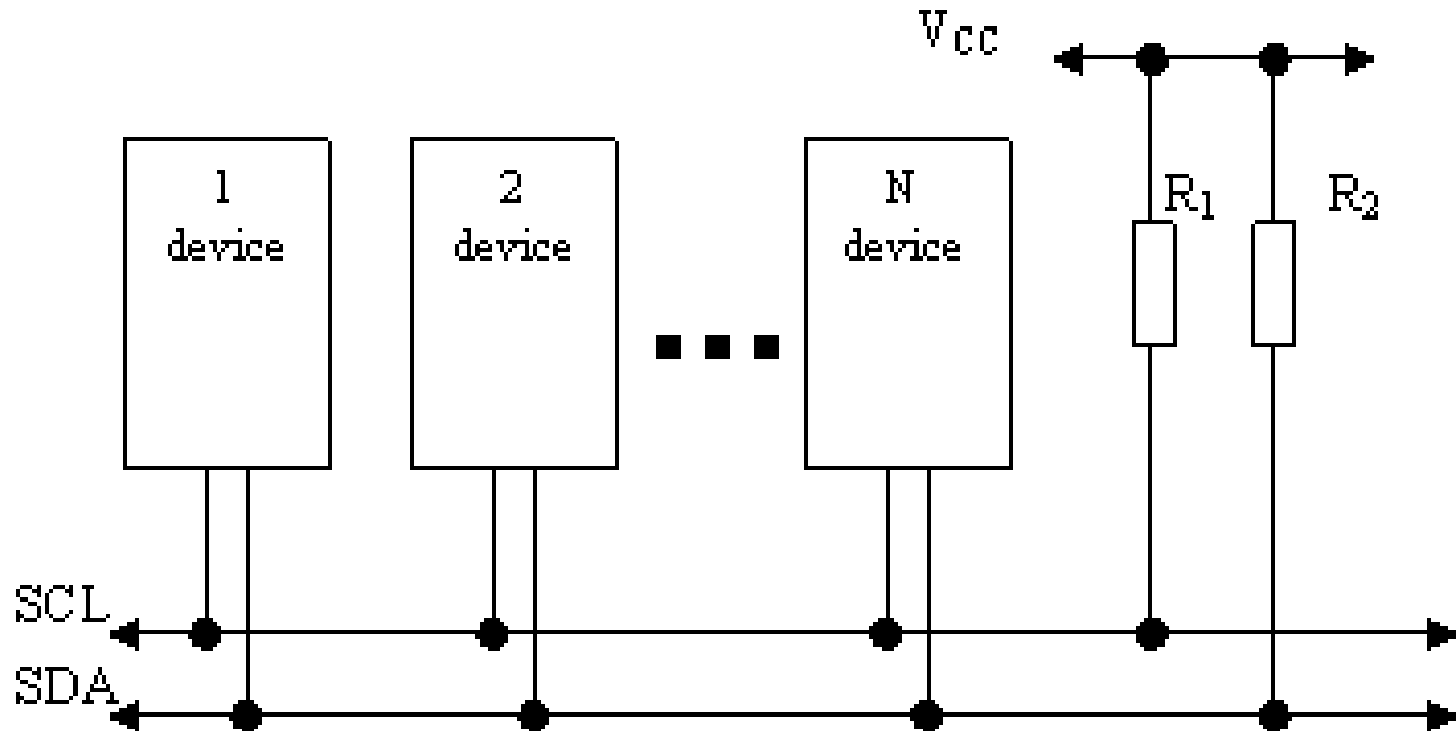
# The Robot *is* the Network

R.A.N. = Robot Area Network™

- P2P net of RISC nodes
  - Each node manages a cluster of transducers
  - This creates truly autonomous sensors (including roll-your-own)
  - 8 MIPs per node x 8 nodes = 64 MIPs
  - 400 Kbits/sec
  - 1 KHz RAN message update rate

# Robot Area Network (RAN)

“The Robot is the Network”



# RAN Nodes

<b>Name</b>	<b>Device</b>	<b>Interface(s)</b>	<b>I/O Devices controlled</b>
Cortex	ATmega168	Host (TTL asynch serial) Optional serial debug output TWI LAN local to TrackBot™	Buzzer Nav Lights (RGB LEDs x 4)
Motor	ATmega168	TWI LAN	Motor speed, direction Tachometry Odometry Motor current draw
Obstacle	ATmega168	TWI LAN	Cliff sensors x 4 Obstacle sensors x 4
Power	ATmega168	TWI LAN	IR Comm Emitters x 4 IR Comm Receivers x 3 Status LED RGB x 1 Power Switch 5V and 3.3V regulators Coulometer Battery charger USB power for Host
Transducer (forward)	Atmega88 or 168	TWI LAN Serial or SPI to transducers	(optional) Sonar, PIR, CMUcam or other user-provided sensors.
Transducer (port)		TWI LAN	
Transducer (starboard)		TWI LAN	
Transducer (aft)		TWI LAN	

# RAN P2P Details

- Non-destructive, bitwise serial arbitration
  - Similar to CAN
  - High value messages always get through
  - 4 byte message in 160 usec
  - ~4 nodes can communicate every msec

# Speed bump – P2P messages

- “It can't be done” implies assumptions
  - Violate the assumption = void the conclusion
  - Atmel docs claim all MT messages in multi-master system must be same length
- We applied ideas from CAN and UDP
  - Add a simple message header
  - Guarantee arbitration resolves in header
  - Voila – variable-length multi-master MT messages

# RAN Packet Format

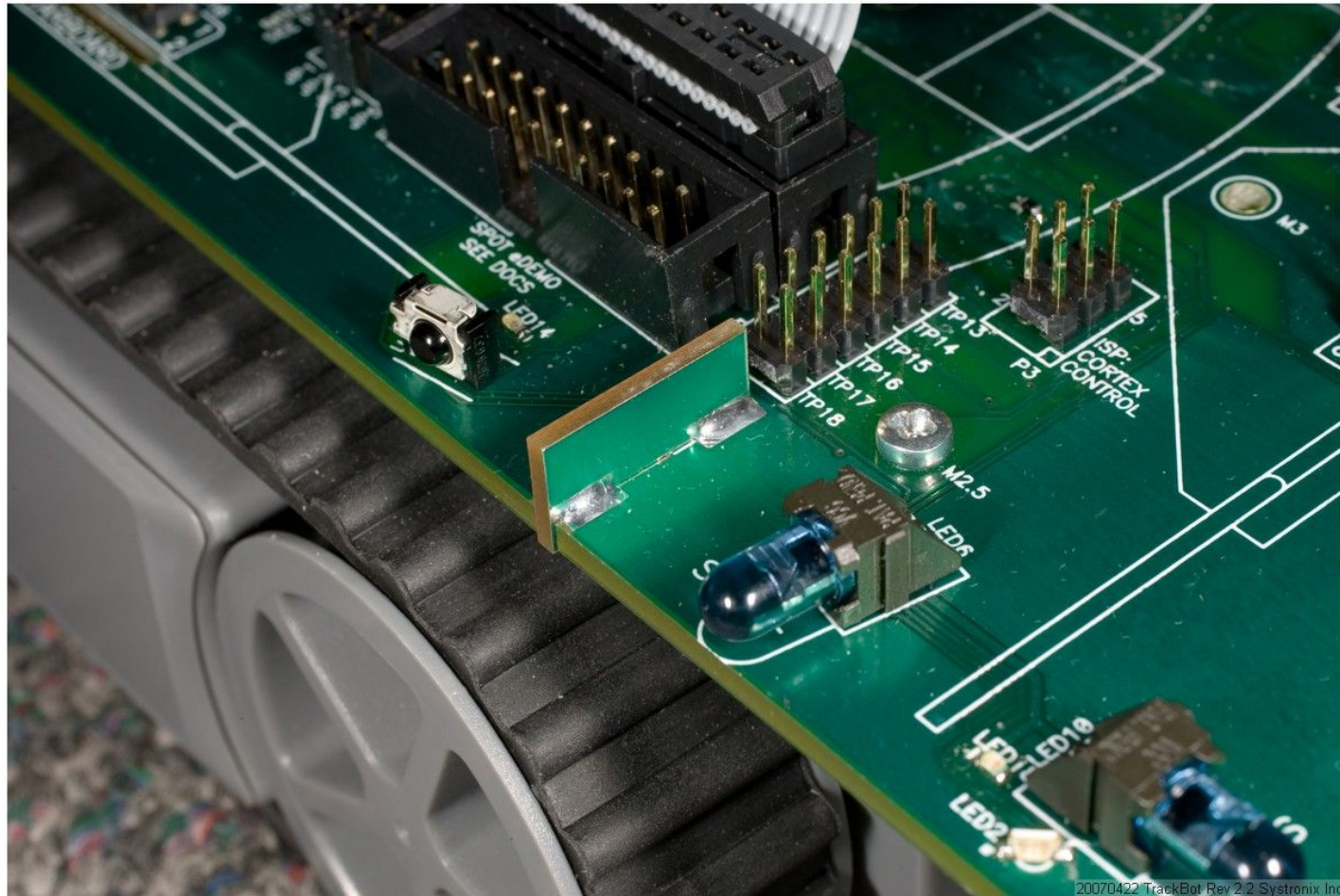
<i><b>TWILAN Master Transmitter Message Packet</b></i>		
<b>TWI Field</b>	<b>Our Field Name</b>	<b>Description</b>
Blue	Blue	Blue
TWI SLA+W	Destination Address	7-bit plus R/W bit = Low, to indicate a Write. Master Transmit messages must always be a Write if the master is sending more than just the destination address. Value can be the General Call Address 0x00 or a specific node.
Blue	Length (4 bits)	Shortest length wins arbitration. Length of zero means there are three bytes of TWILAN packet header and no TWILAN data (e.g., a query message of some type)
	Type (4 bits)	See Type table
Blue	Blue	Blue
Data byte	Source Address	7-bit TWI - plus padding bit (lsb=0)
Green	Green	Green
Data byte	Data byte 0	1 <sup>st</sup> byte of TWILAN data, out of N-1 bytes total
Green	Green	Green
Data byte	...	Data byte N
Green	Green	Green
Data byte	Data byte N-1	Max of 15 bytes of data payload if length == 1111
Notes:		
1- Blue cells denote required header. Green cells are optional data payload.		
2- msb of a byte is sent first		

<b>TWILAN Master Transmitter Message Type Field</b>		
Emergency	0000 (command)	It's, well, an emergency, e.g., the battery is extremely low, so shut down all non-essential services and get to the charging station ASAP. Or else the TrackBot™ dies!
	1000 (response)	OK, here's what I am doing about that emergency. Can't think of an example.
Action	0001 (command)	Do something with the data to follow, e.g., change the state and speed of a motor, and here's the data to do that.
	1001 (response)	OK, that command is complete. Typically doesn't return data, just indicates that some task is done, e.g., the motors are now running at the level you requested. This ACK might not follow for some time, e.g., if the command was to find a charging station, that could take several minutes.
Data Query	0010 (command)	I'm requesting that you make a subsequent response. Data payload could include further query options, e.g., tell me the value of thermocouple #132, or how far the left and right tread motors have travelled since this session began.
	1010 (response)	This message is a response to an earlier query, e.g., here's the total centimeters travelled by both motors, as two 32-bit integers.
Enumerate	0011 (command)	Tell me about yourself... send me all your tagging information, firmware version, etc.
	1011 (response)	Here's my enumeration response, which could be quite lengthy, if it includes tagging information. ??? Enumeration response might always be a segmented message.
Segmented Message	0100 (write)	A way to send really long data such as a tagging XML file, musical tone sequence, or some image data. For example, this message is part of a new executable being sent to the destination. Will probably include a sequence value and finally a checksum. Refer to <i>Updating TWILAN nodes</i> .  Payload byte 0 is the fragment number, with values [1..255] indicating the number of fragments. Fragment value of 0 indicates that this is the last fragment. For example, a message with 3 fragments would have them numbered 1,2 and 0, where 0 is the last fragment.
	1100 (ACK)	Typically no data payload per se, but could ACK with the fragment number as the only payload.
	0101 (command)	Administrative type messages which are not really specific to a given node. Examples: Set verbosity of responses, set whether a response (or not) is required to all actions.
Admin	1101 (response)	Might just be an ACK.
	0111 (write)	Philips reserves address 111 xxxx so we thought we'd do the same for types, on general principles.
Reserved	1111 (ACK)	This pair might be used to force the hardware into some special test or update mode.
	Notes:	

# Message Types

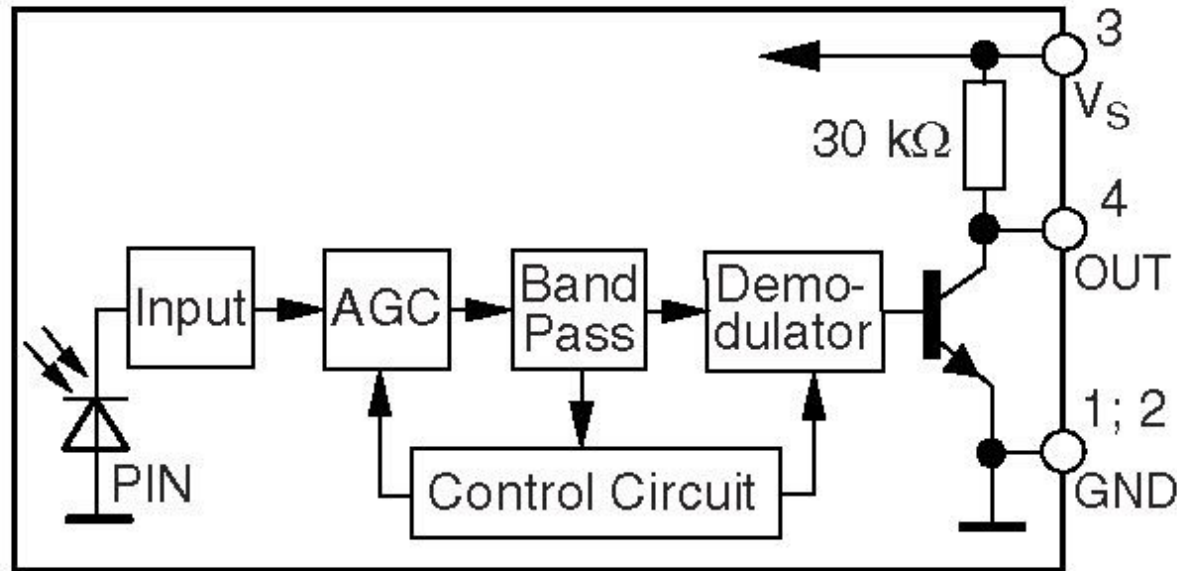
- Emergency
- Action
- Data Query
- Enumeration
- Segmented
- Admin
- Reserved

# Multi-function IR



# Multi-function IR Emitters/Sensors

- Use low-cost but programmable electronics
  - High power IR emitters and integrated receivers (diagram below)

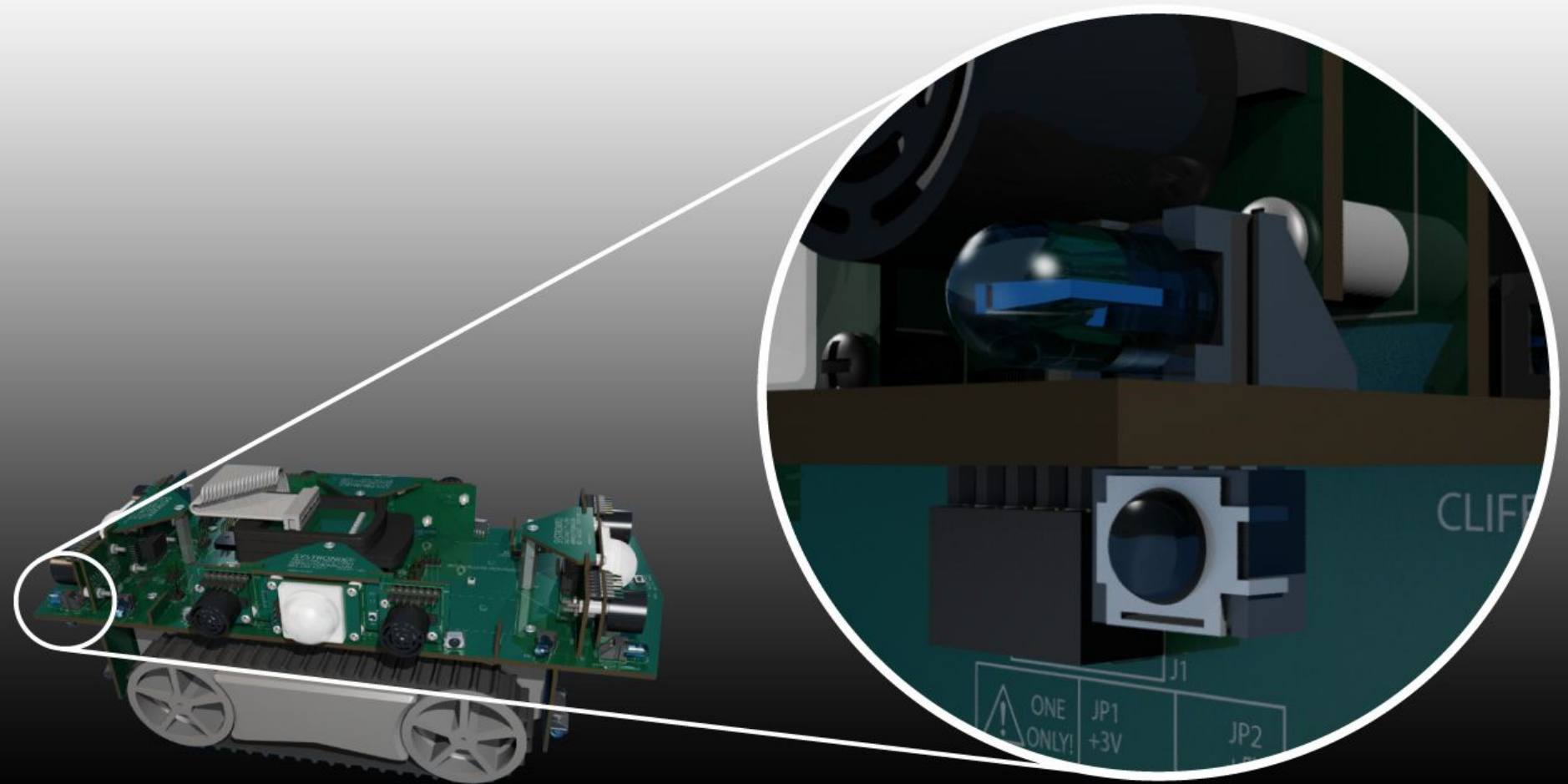


16839

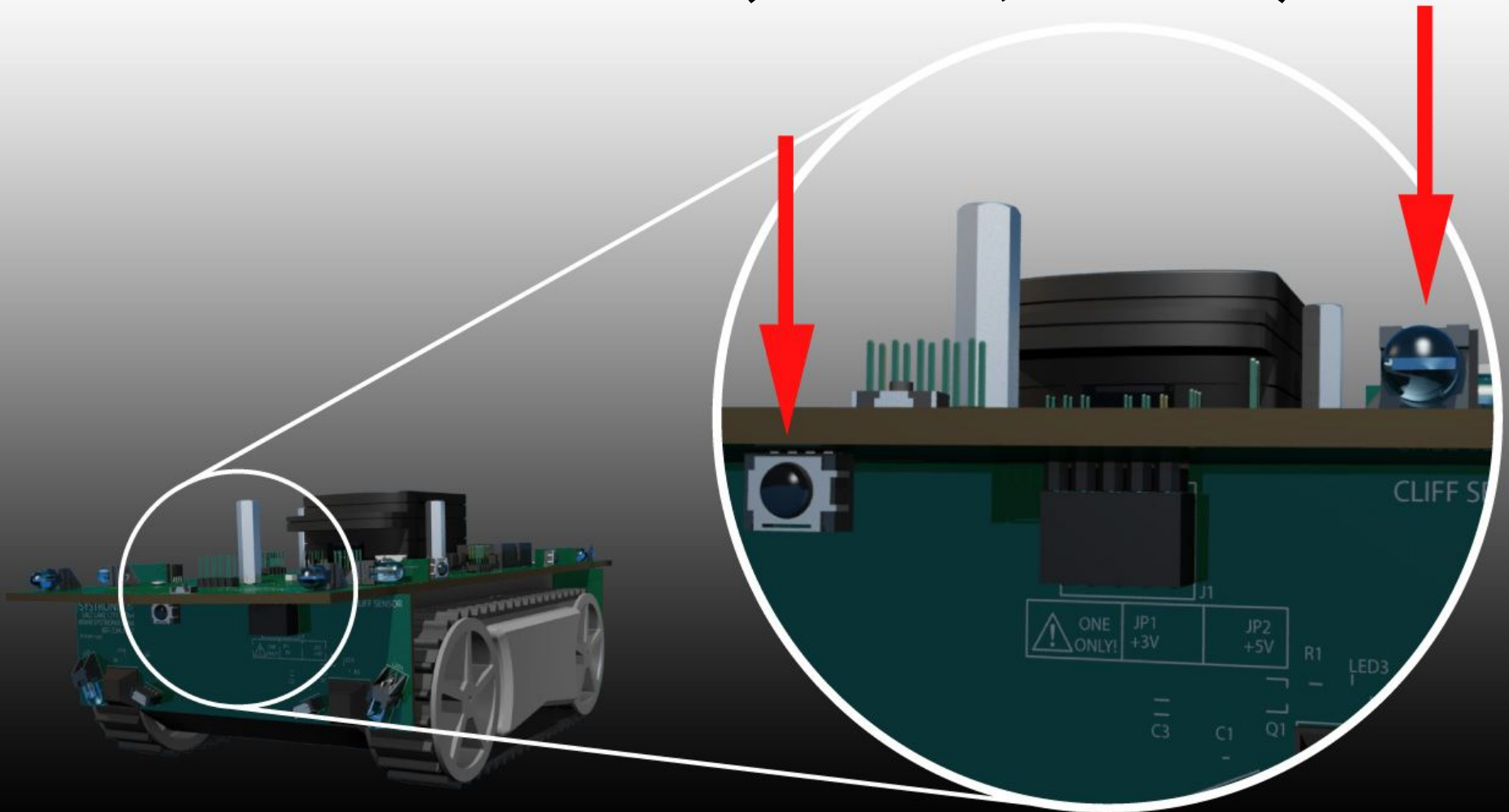
# 5-function IR Emitters/Sensors

- With intelligent protocol firmware, we can accomplish simultaneous(!)
  - obstacle avoidance
  - crude rangefinding (0, -3 dB, -10 dB now)
  - Beacons per emitter
  - communication
  - spatial awareness
- All this is not currently possible with more costly sonar or IR rangefinder modules

# Forward Sensors x 2

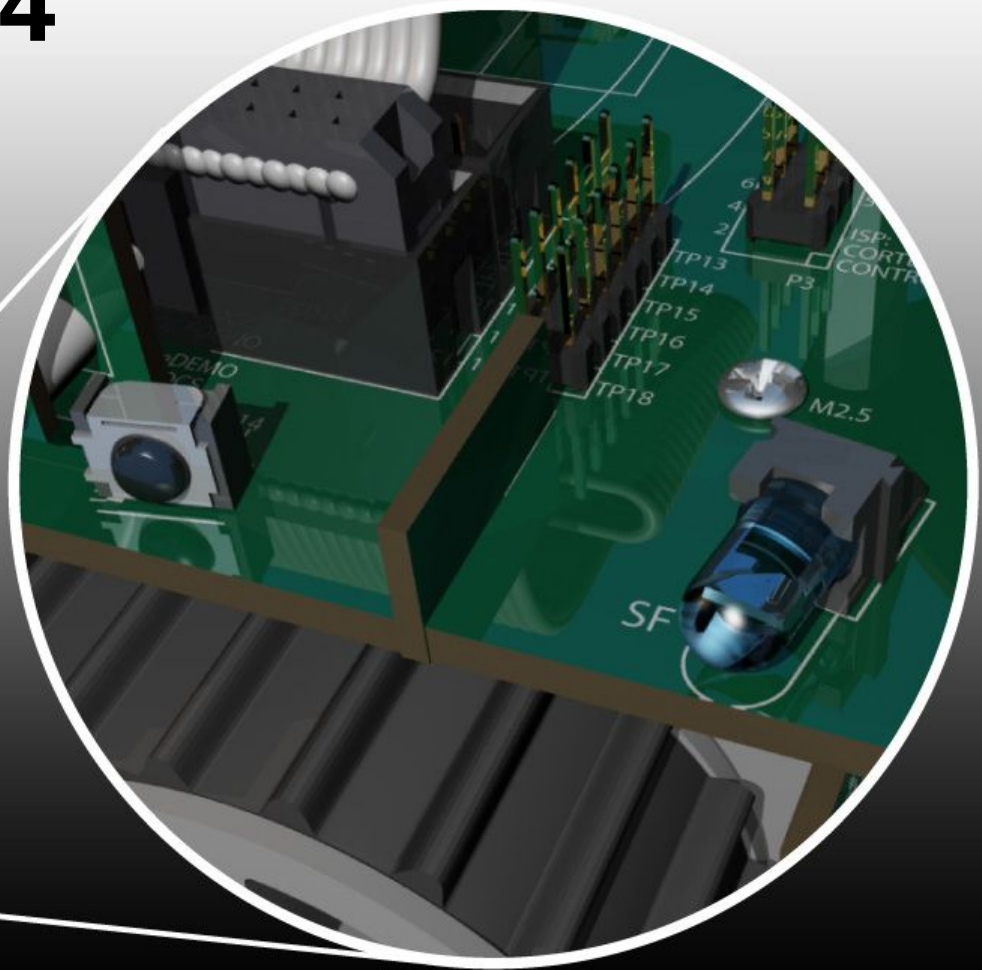
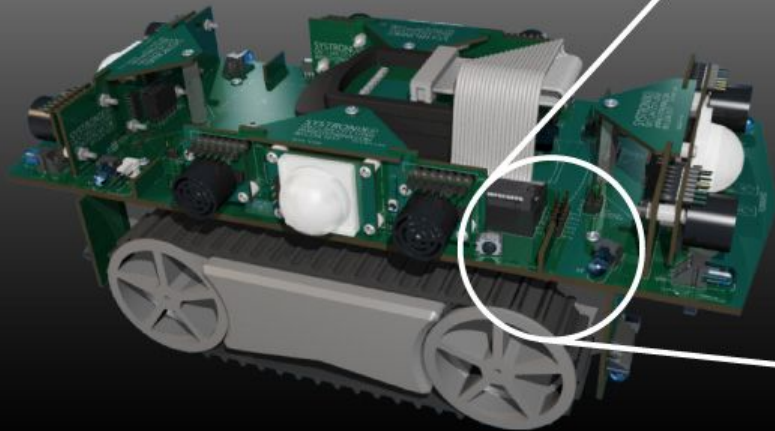


# Aft Sensors (2 Xmt, 1 Rcv)

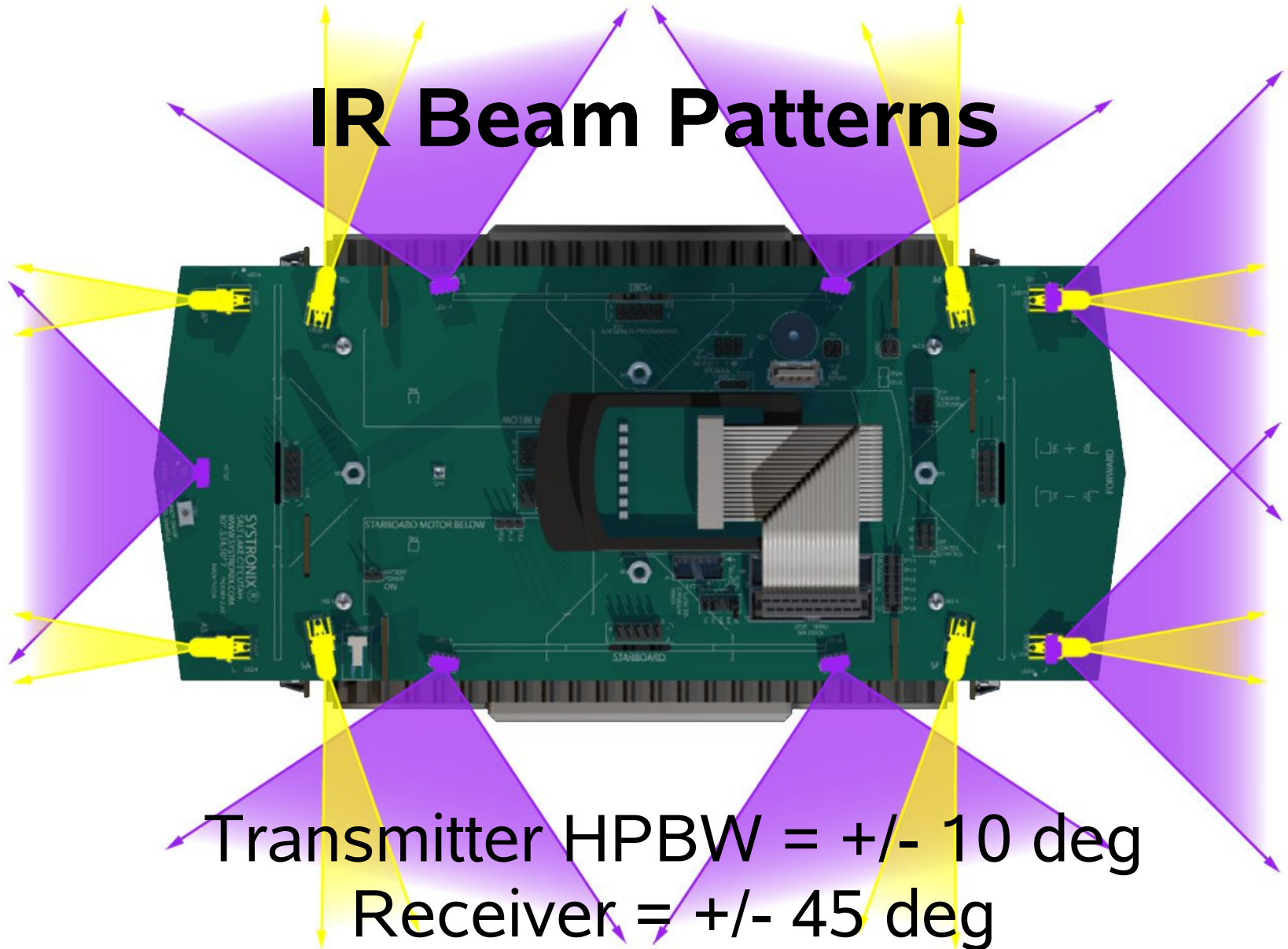


# Side Sensors x 4

- 100 mm focal point



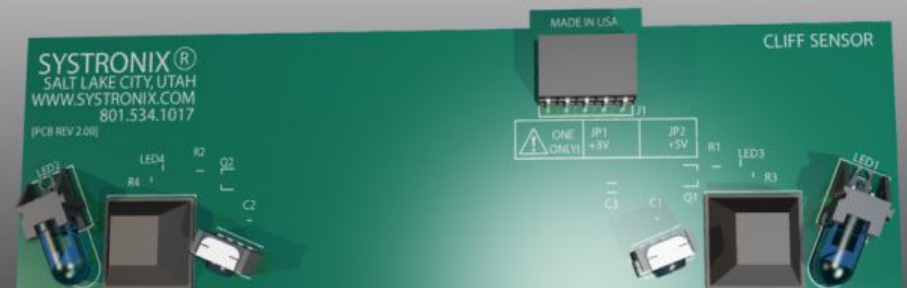
# IR Beam Patterns



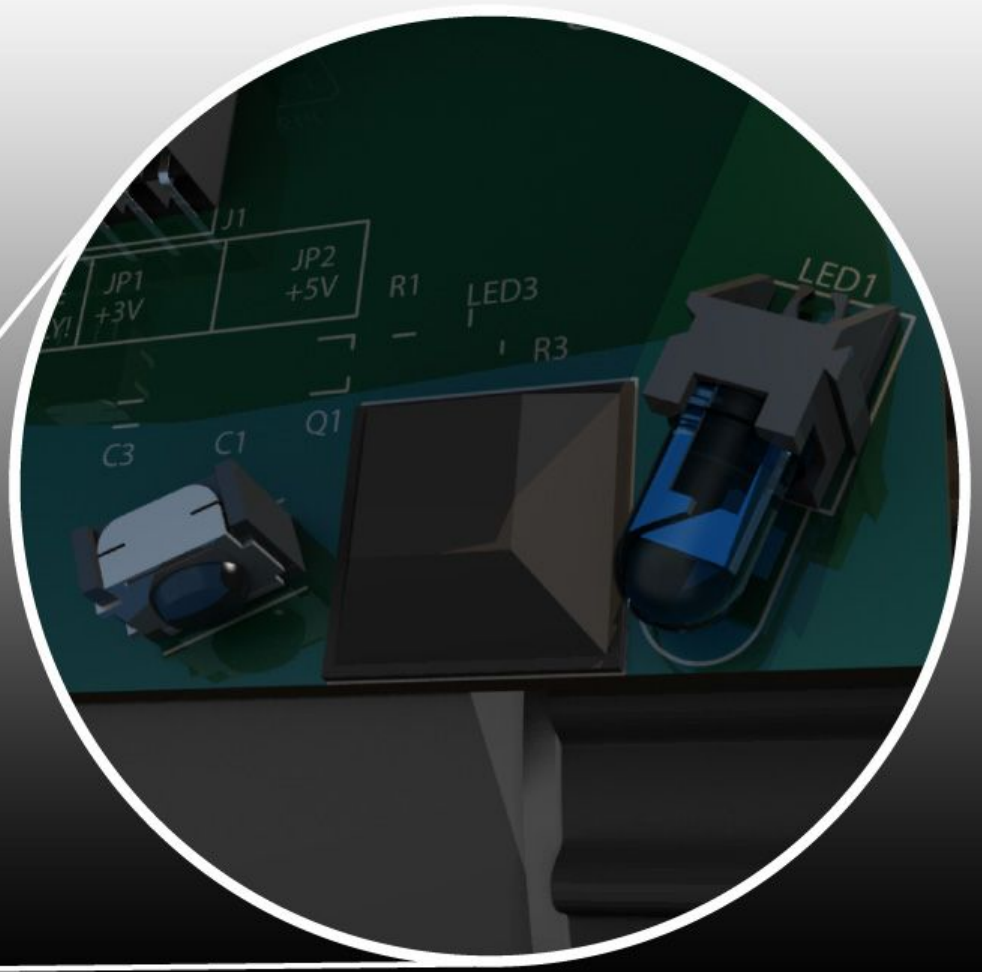
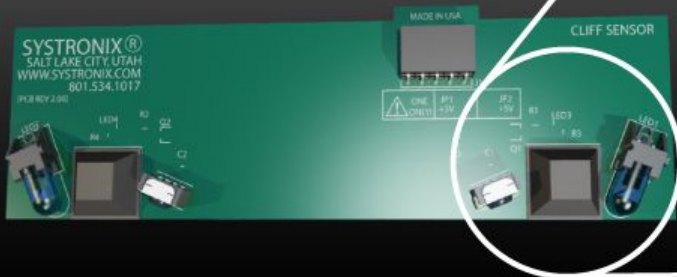
Transmitter HPBW = +/- 10 deg

Receiver = +/- 45 deg

# Cliff Sensors (optional) fore and/or aft



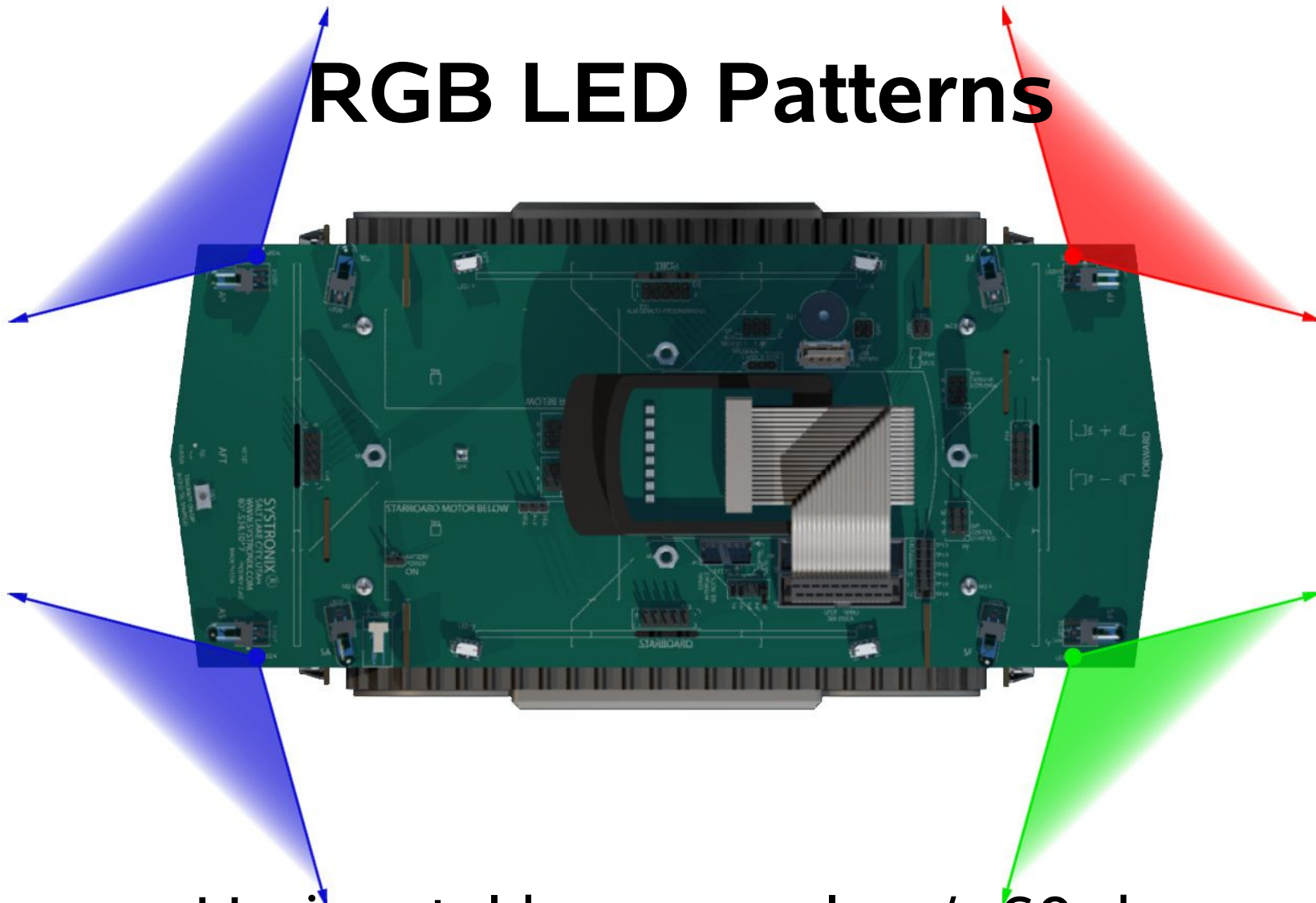
# Cliff Sensor IR Details



# RGB LEDs on each corner

- Video analysis of behaviors
- Visual display of spatial orientation, like aircraft and boats
- Teams and games
- Visual cues as part of user interface
- It just seemed like a good idea

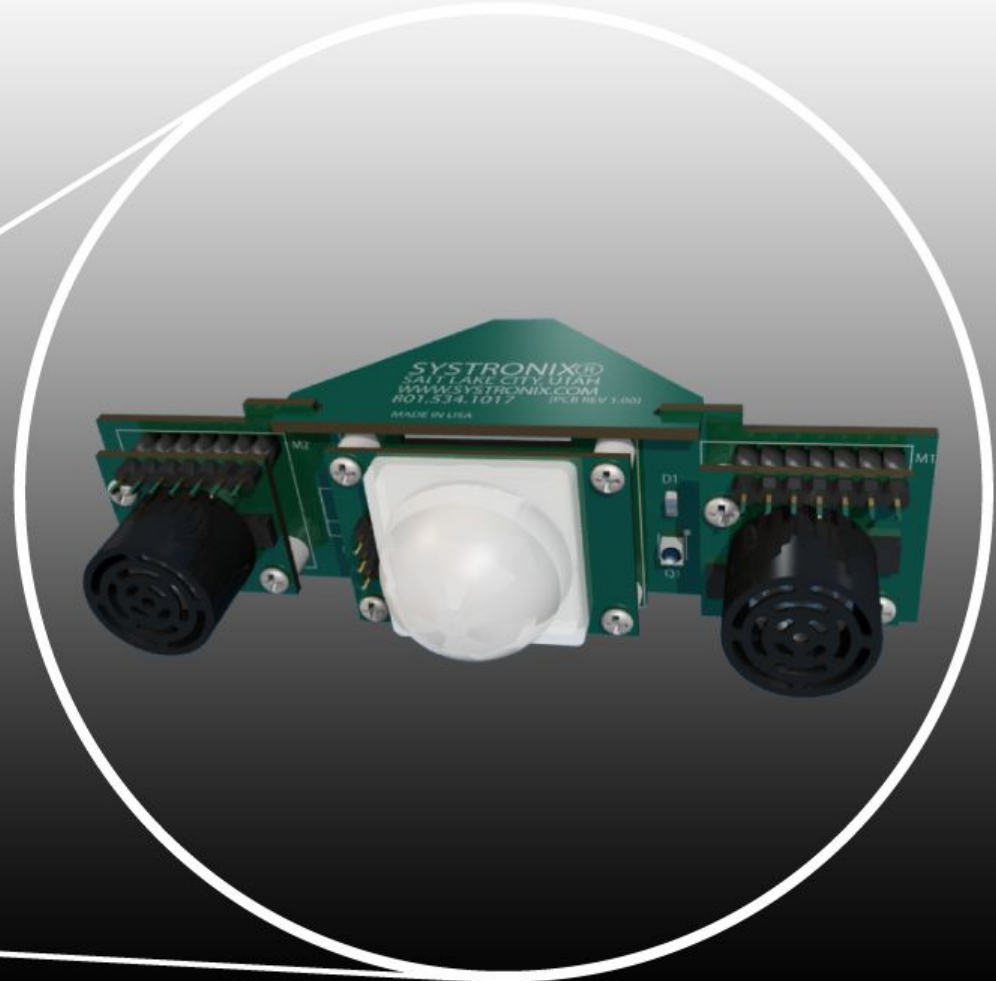
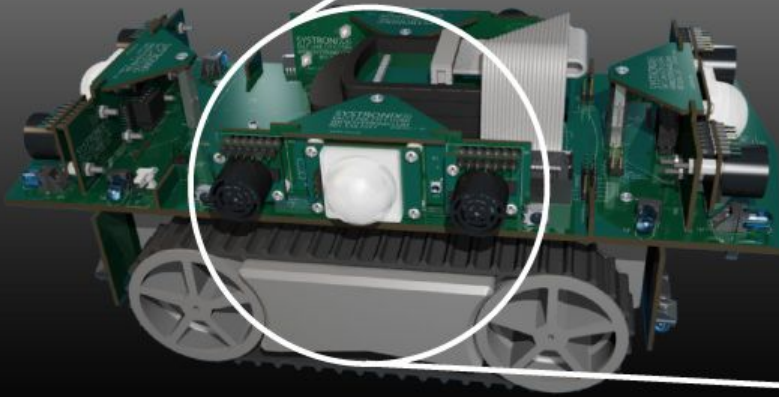
# RGB LED Patterns



Horizontal beam angle +/- 60 deg

# Transducer Station

- Sonar x 2
- PIR
- Ambient IR & Vis



- 7.2V 15 Watt-hour NiMH

Battery Pack

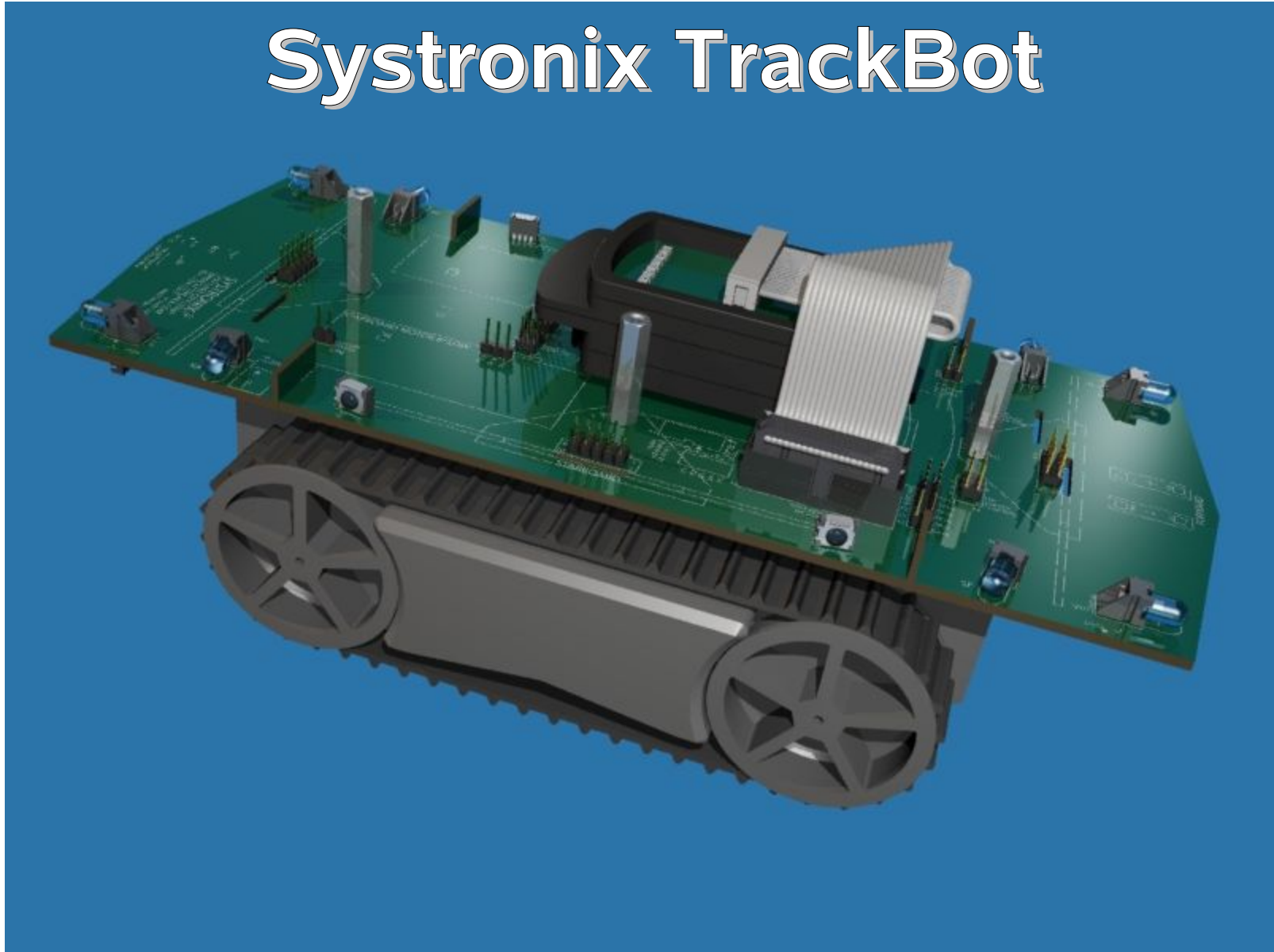
# Results

- Tracked chassis
  - drives over common obstacles
  - Appears to be sufficiently durable
- Multi-function IR sensors built on
  - 8-device array is feasible & affordable
  - Sensors look very usable in practice, even with simple software
  - User-expandable Transducer stations

# Results

- Real-time Robot Area Network
  - 160 usec per small message
  - 1 msec loop time easily achieved
- Battery Life
  - > 2 hours of heavy use, 8 hours light use
  - > 1000 hours of standby (200 uA)
- HW in production
- SW still in development, very usable now

# Systronix TrackBot



# What's Next?

- Charging Tower, solar charger
- Complete 802.15.4 and Zigbee support
- Prototyping Transducer Station
- More Driver and Host API Code
- Projects at [java.net](http://java.net) in robotics community
- Integrate with Greenfoot and StarLogo?
- Swarm research – NCSU is the first
- CMU Cam, speech, etc

# Videos & Short Demo

- Live demo with small swarm Fri at 2PM
- Mixture of tele-operated (via 802.15.4) and autonomous TrackBots

# References

- [www.trackbot.systronix.com](http://www.trackbot.systronix.com)
- YouTube videos  
<http://www.youtube.com/profile?user=JavaRobotics>
- [www.jstamp.com](http://www.jstamp.com)
- [www.java.net](http://www.java.net) - robotics community  
Source code mentioned here:  
<https://spottrackbot.dev.java.net/source/browse/spottrackbot/trunk/>  
<https://trackbotcode.dev.java.net/source/browse/trackbotcode/trunk/>
- [www.sunspotworld.com](http://www.sunspotworld.com)

# References

- 3D models by Daniel Fish & Blender  
[daniel@cs.utah.edu](mailto:daniel@cs.utah.edu)
- Demo code by Shawn Silverman, Qindesign  
[shawn@pobox.com](mailto:shawn@pobox.com), <http://tynamo.qindesign.com>