BRADLEY University

COOPERATIVE CONTROL OF HETEROGENEOUS MOBILE ROBOTS NETWORK

Gregory Bock, Brittany Dhall, Ryan Hendrickson, & Jared Lamkin

Project Advisors: Dr. Jing Wang & Dr. In Soo Ahn

Department of Electrical and Computer Engineering

April 26th, 2016

Outline

I. Introduction
II. E-puck – Brittany
III. Kilobot - Jared
IV. QBot 2 – Ryan & Greg
V. Summary & Conclusions

I. Introduction

Objectives

- Design and Experimental Validation of Cooperative Control Algorithms
  - Sensing/communication between robots
  - Implementation of local flocking control algorithms
  - Implementation of local formation control algorithms

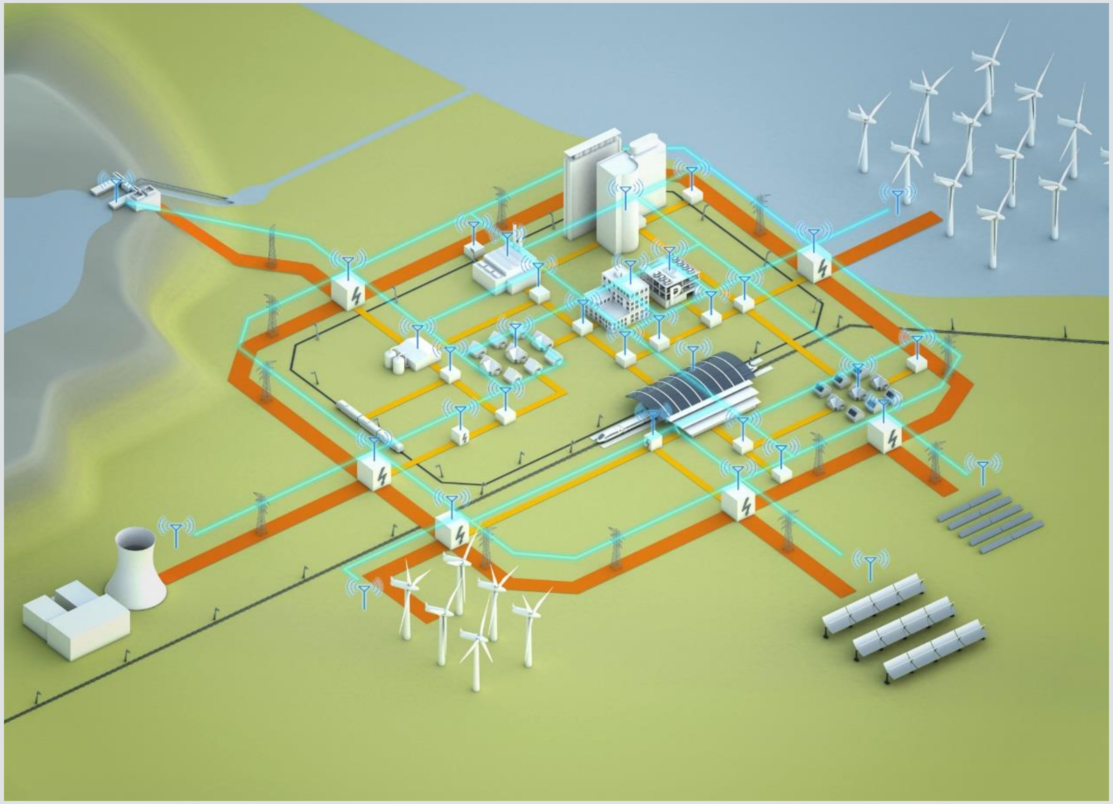
Project Background

- Cooperative systems found in nature
  - Flock of birds
  - School of fish
  - Swarm of insects

http://www.huffingtonpost.com/2013/10/07/plane-hits-bird-ohare_n_4058132.html

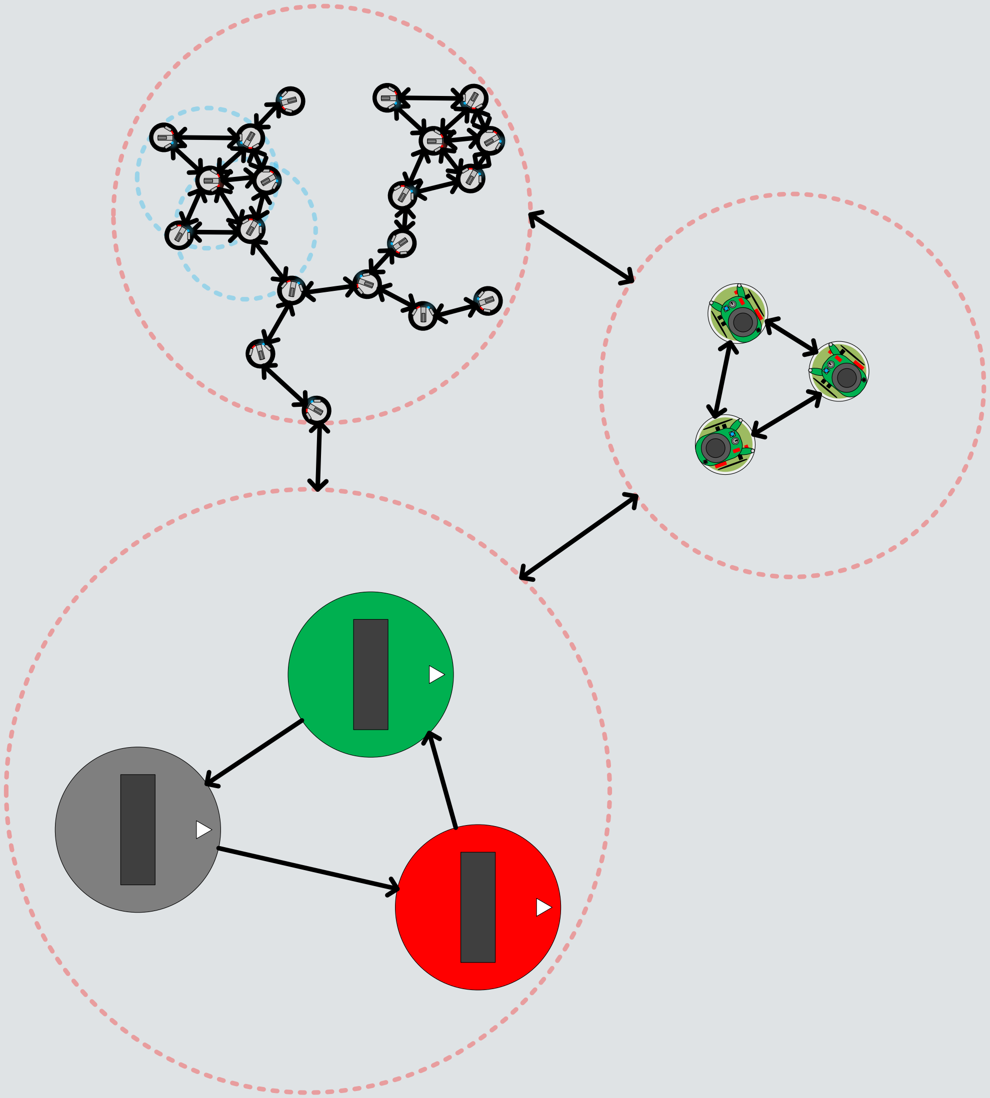
Possible Applications

- Cooperative systems found in engineering
  - Smart Grid
  - Sensor Network
  - Traffic Network

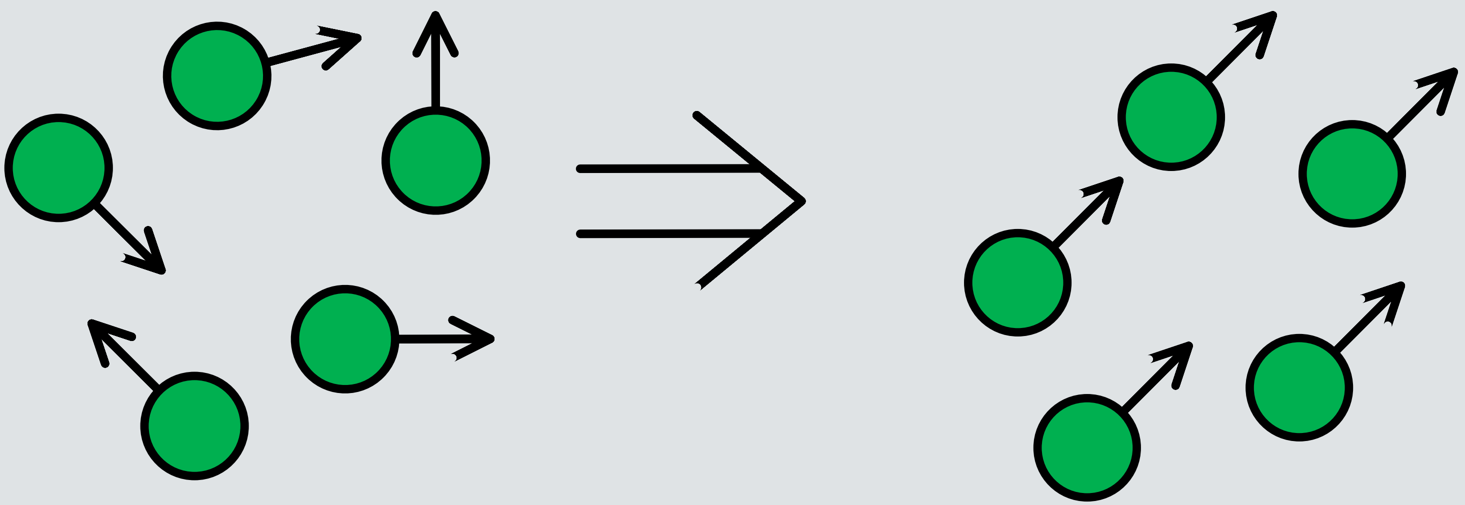


http://www.siemens.com/press/en/events/2012/corporate/2012-06-wildpoldsried.php

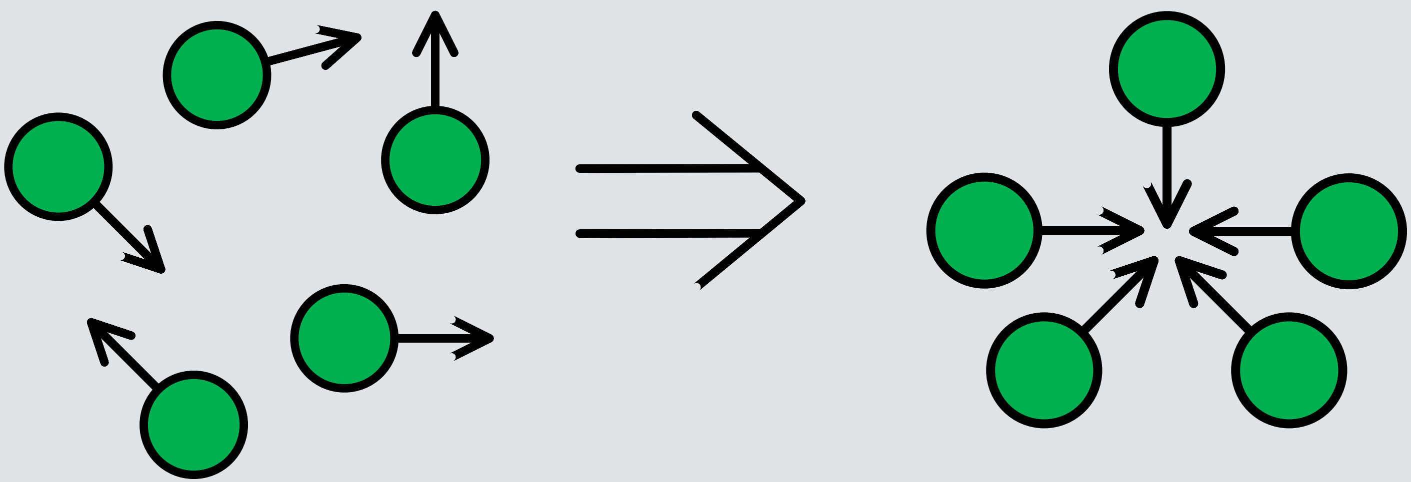
Heterogeneous Groups



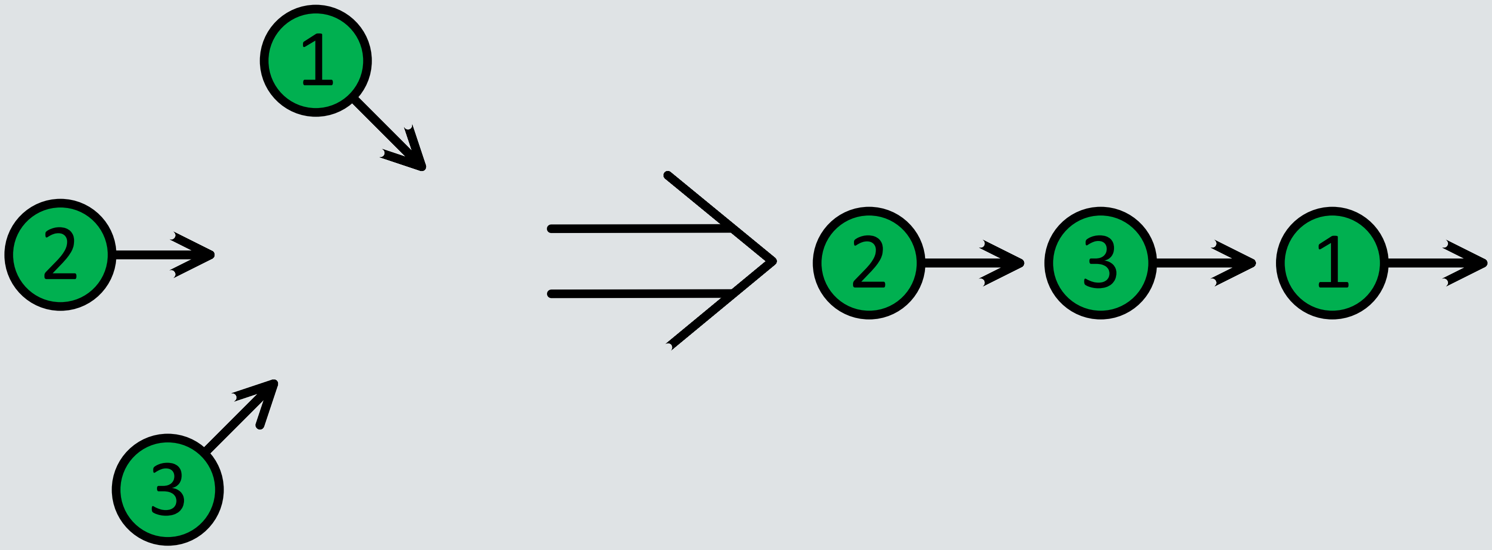
Heading Alignment



Point Consensus



Following



Design Constraints

- Must overcome limited communication among networked robots
- Must overcome limited sensing capability of robots
- Must overcome system uncertainties

Test Platform – Kilobot

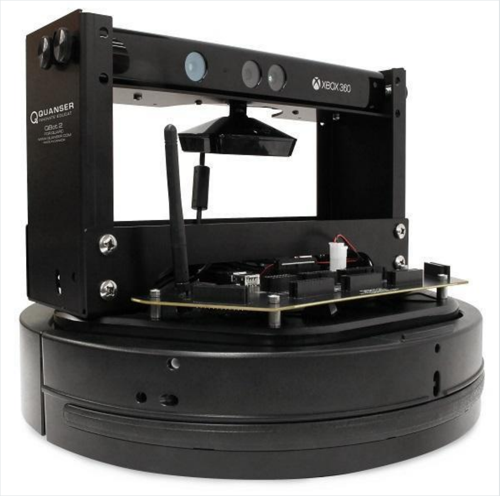
- Diameter of 3.3 cm
- Two differential vibration motors
- IR transmitter and receiver (7 cm range)
- Ambient light sensor

Test Platform – E-puck

- Diameter of 7 cm
- IR transmitter and receiver ring (25 cm range)
- On-board CMOS camera
- Bluetooth 2.0
- dsPIC 30F6014A on-board computer

Test Platform – QBot 2

- Open-architecture autonomous ground robot
- Xbox 360 Kinect
- Kobuki robot base
- Gumstix DouVero Zephyr on-board computer



II. E-puck – Brittany

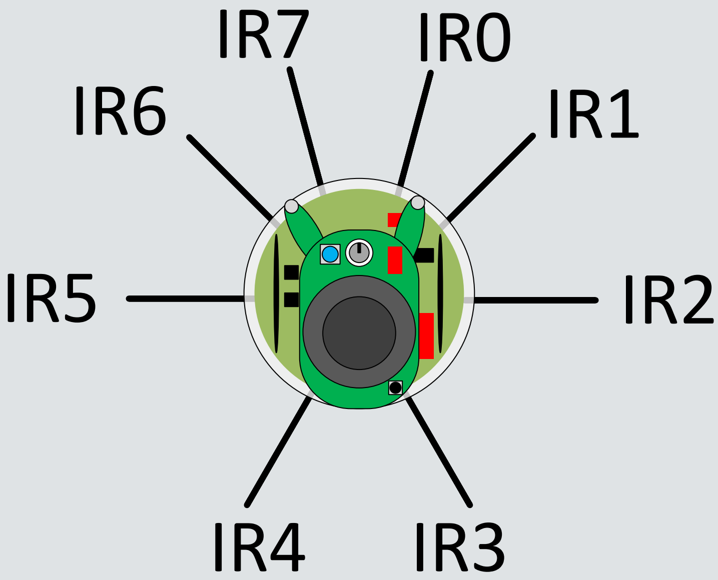
Work Accomplished

- Software & hardware implementation
- Object detection/following
- Odometry
- Vicsek Model
- Fix battery issues

https://www.cyberbotics.com/item?id=8

Infrared proximity sensors

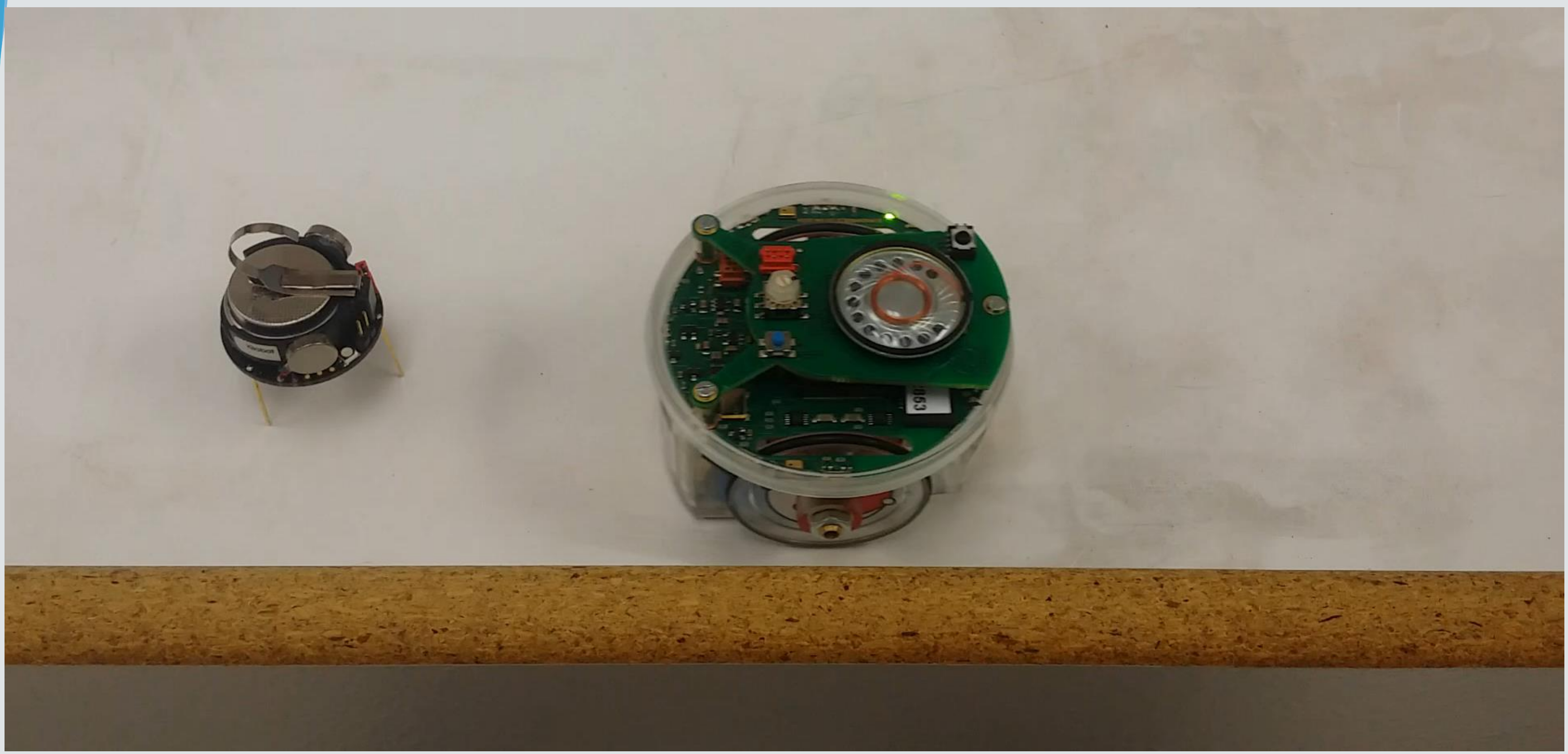
- 8 infrared proximity sensors
  - Composed of two parts -IR emitter & photo-sensor
- Can detect objects within 4 centimeters



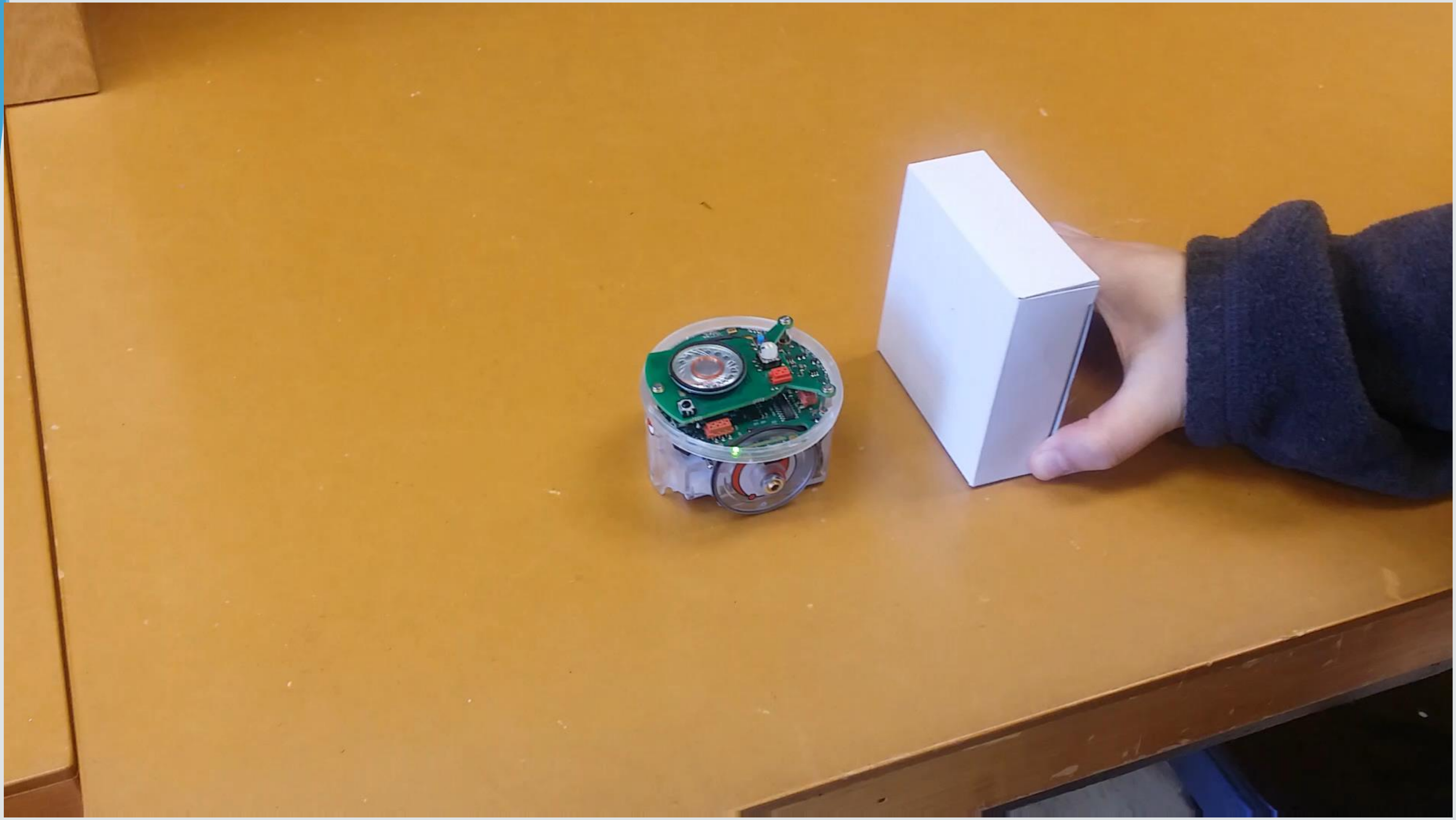
Object Detection and Following

- Proximity sensors -> detected distance
- Compare with true specified distance
- Velocity = gain*(specified distance – detected distance)

Object Detection



Object Following



Odometry

- Using odometry the E-puck can compute their position and orientation

$$\triangle x = \triangle S * cos\left(\theta(k) + \frac{\triangle \theta}{2}\right)$$

$$\triangle y = \triangle S * sin\left(\theta(k) + \frac{\triangle \theta}{2}\right)$$

$$\theta(k+1) = \theta(k) + \frac{\triangle \theta}{3}$$

- $\triangle S$ – average change in steps of both left and right motors
- $\triangle \theta$ – change in the angle of the agents heading

Vicsek Model

$$\theta_i(k+1) = \frac{\theta_i(k) + \sum_{j=1}^{n} \theta_j\,(k)}{n+1}$$

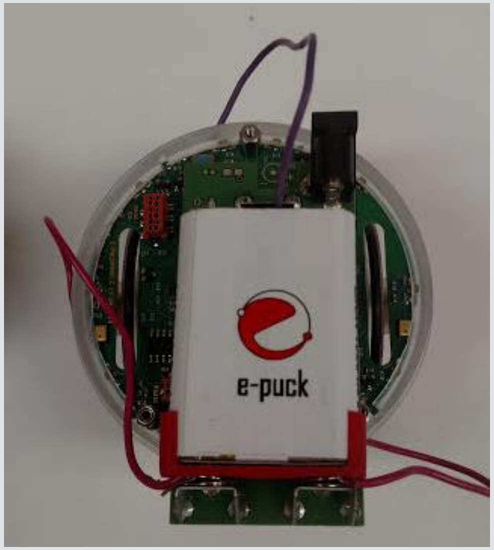
- $\theta_i(k+1)$ - Next heading of agent
- $\theta_i(k)$ - current heading of agent
- $\sum_{j=1}^{n} \theta_j(k)$ - sum of all neighboring agents at time k
- n - number of neighboring agents

Vicsek Model

E-puck Battery Problem – Solution

Original Design

•Bad connection between positive and negative terminals from battery to E-puck



Solution #1

•Added an addition on top of E-puck, for better connection to terminals

Solution #2

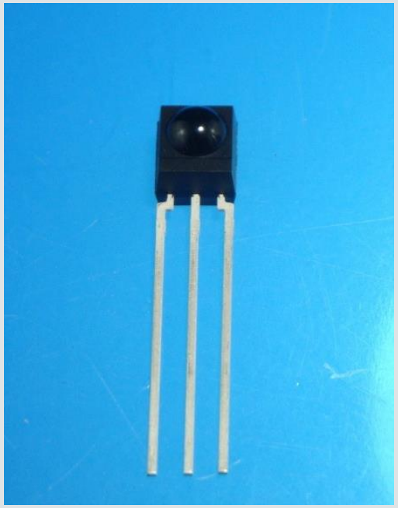
•Resoldered positive terminal

Testing communication between E-puck and Kilobot

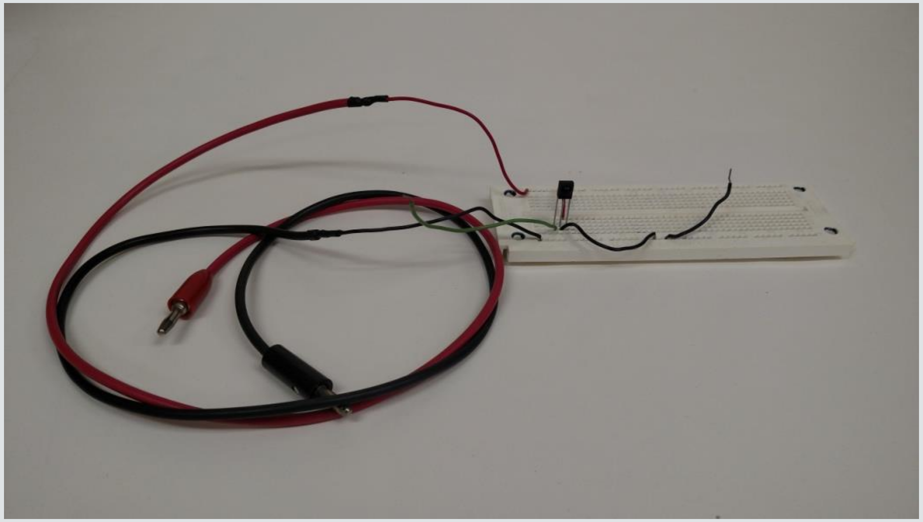
- Tested E-puck communications with infrared receiver connected to oscilloscope initially, followed by testing with Kilobot
- Verified E-pucks sent message with correct protocol
- Verification of communication between E-puck and Kilobot would be accomplished by observing change in LED from red to green

Infrared Receiver Circuit

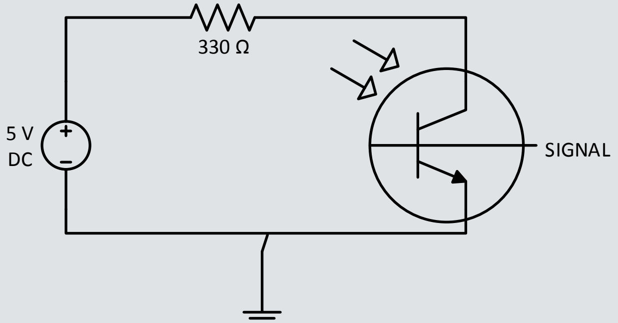
38kHz Infrared Receiver Module



Infrared Receiver Circuit



Used a 5V supply & oscilloscope to view the signals

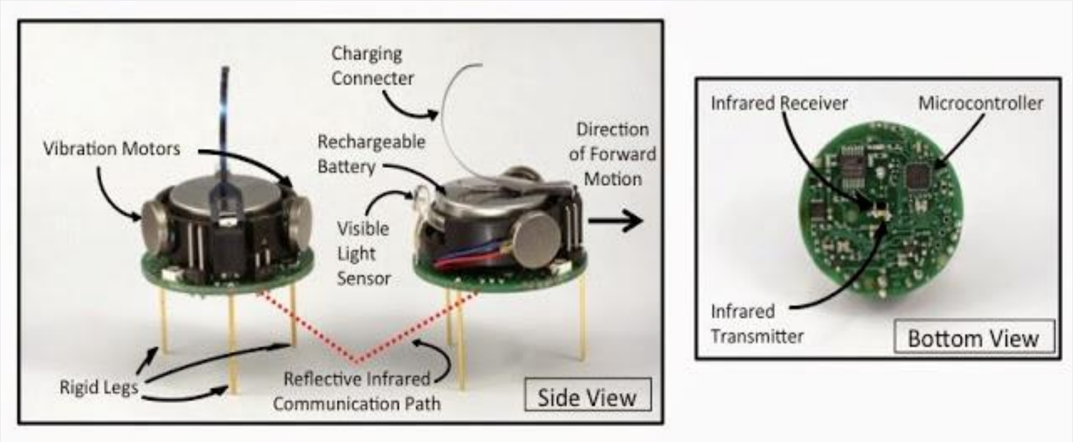


http://www.ebay.com/itm/like/141932065528?lpid=82&chn=ps&ul_noapp=true

III. Kilobot - Jared

Kilobot

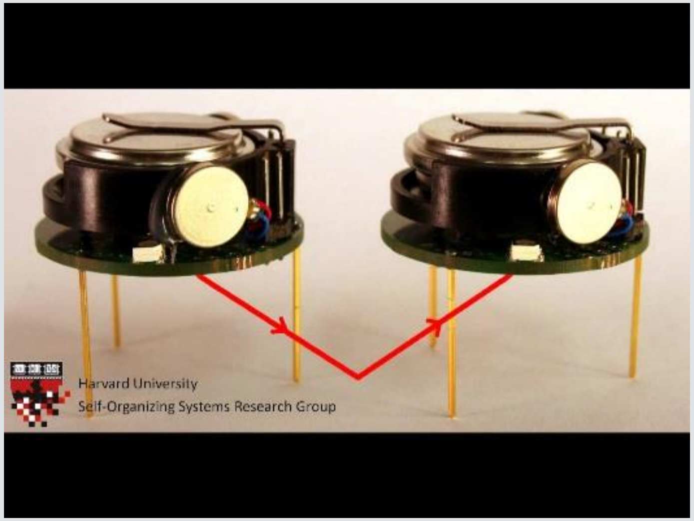
- Atmega 328 (8-bit @ 8 MHz)
- 32kB flash, 1kB EEPROM, 2kB SRAM
- 2 vibration motors
- IR LED and receiver
- Ambient light sensor



https://lh3.googleusercontent.com/-g2lSChnX4DI/U-1VyxOKwsI/AAAAAAAAL9A/3mi89VoBBfs/s640/kilobot-closeup-overview.jpg

How Kilobots Communicate

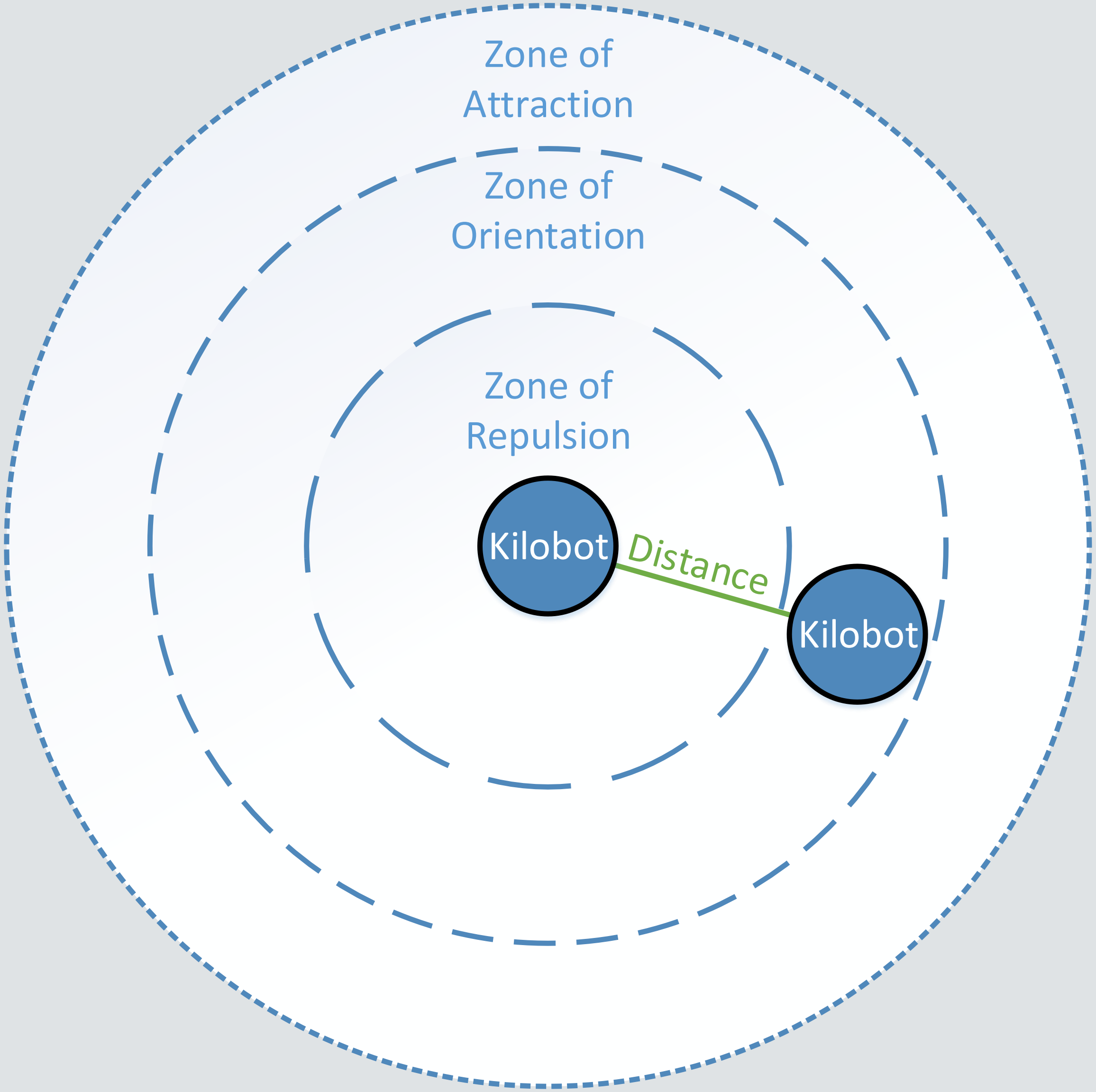
- Use infrared light
- Measures light intensity to calculate distance
- Messages are sent every 200 milliseconds



https://i.ytimg.com/vi/ISMwLCFwgK4/maxresdefault.jpg

Color Synchronization Video

Kilobot Movement: Orbiting



Multiple Agent Orbiting

Simple Localization: Gradient

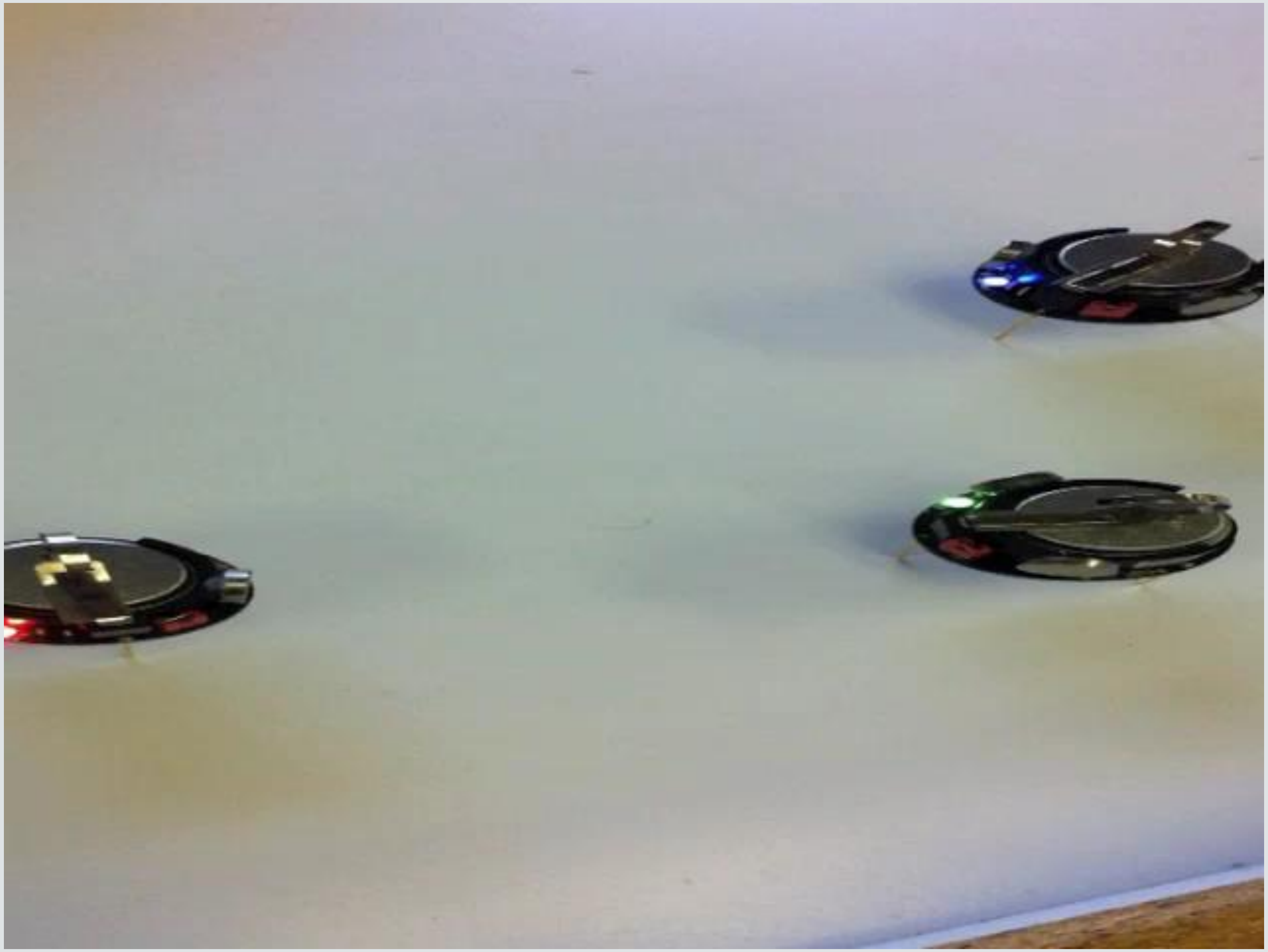
- Can determine how many agents are displaced from a specified agent
- Individuals receive gradient values from local agents until a buffer is full
- Smallest value in buffer is incremented by 1, which becomes agent's gradient value

Gradient

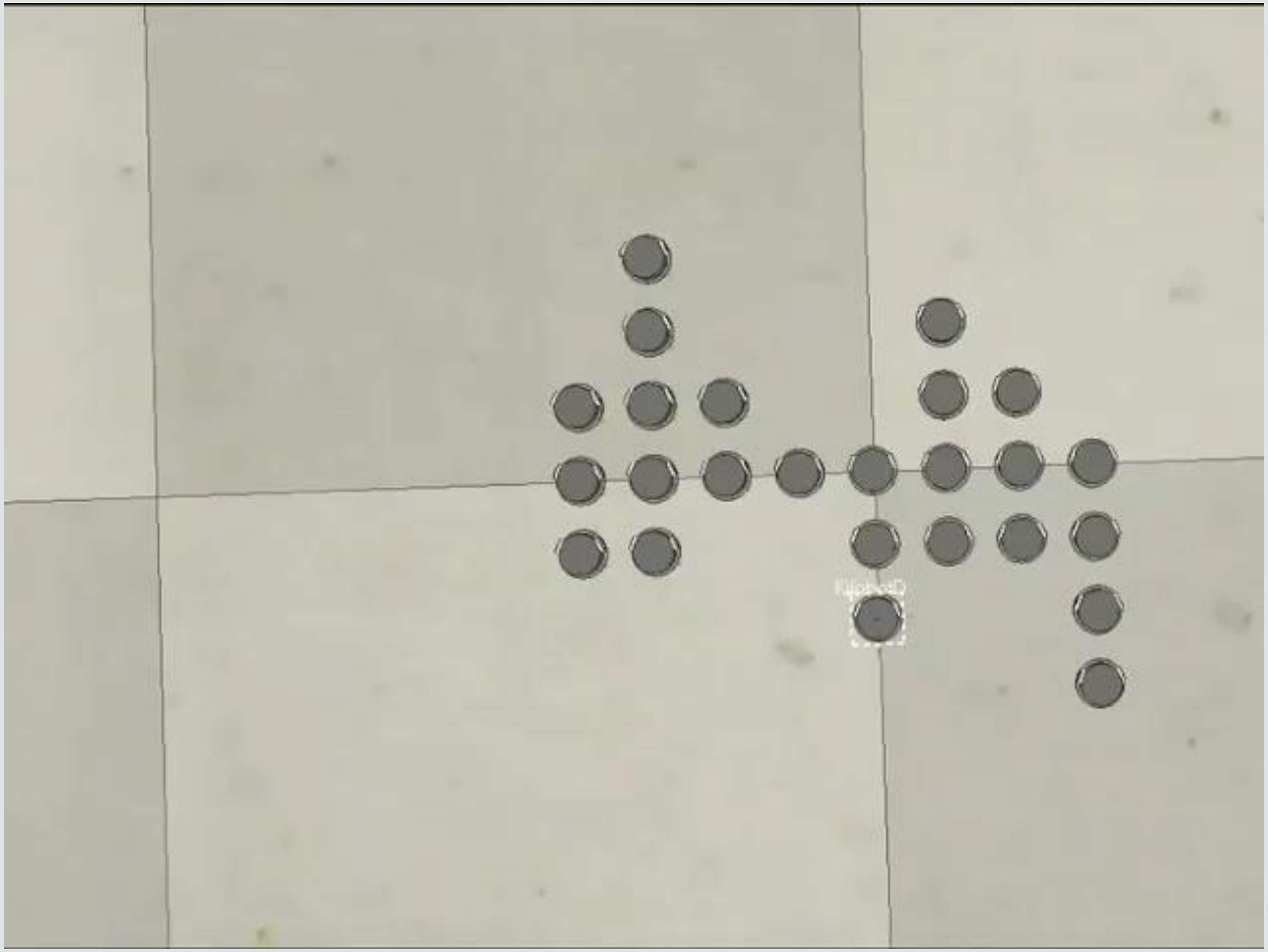
Advanced Behaviors

- By combining gradient, orbiting, and/or light detection more advanced behaviors can be achieved such as:
- **Fixed-point consensus**: Kilobots converge to a fixed-point
- **Edge following**: Kilobots orbit multiple stationary agents
- **Follow-the-leader**

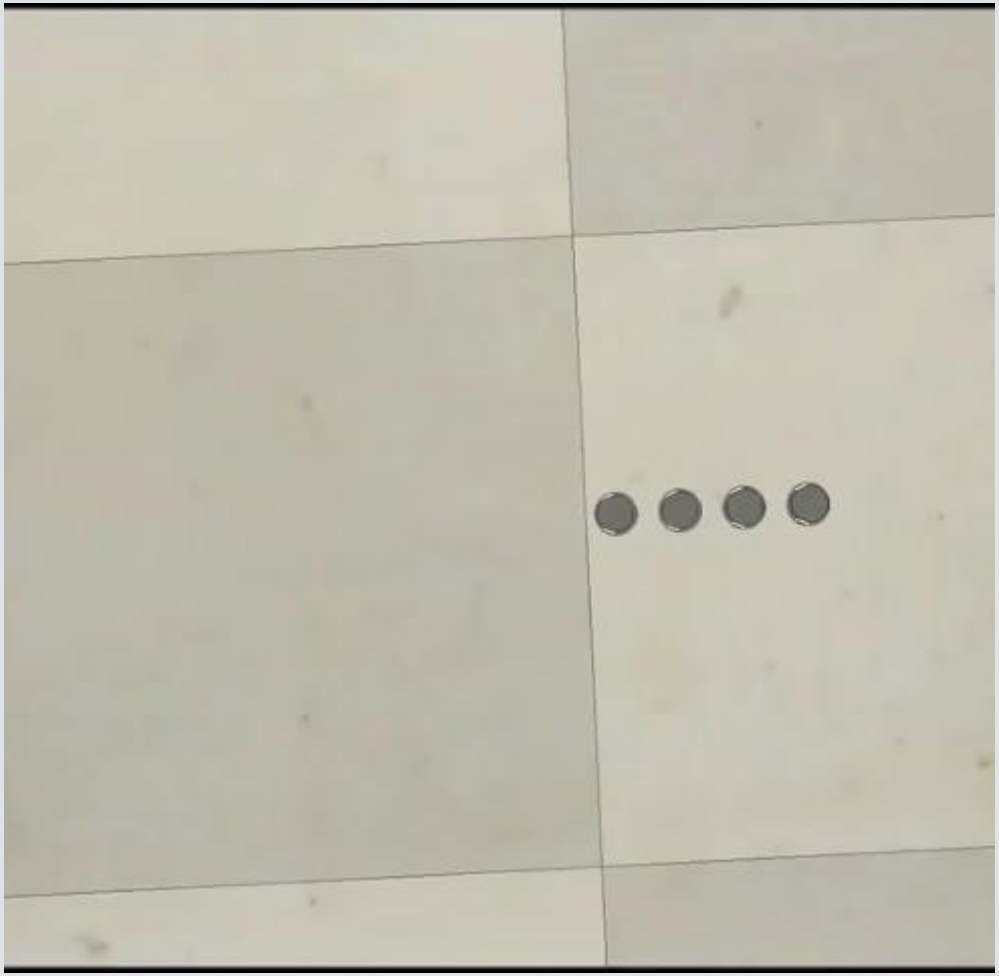
Fixed-Point Consensus



Edge-Following



Follow-the-Leader

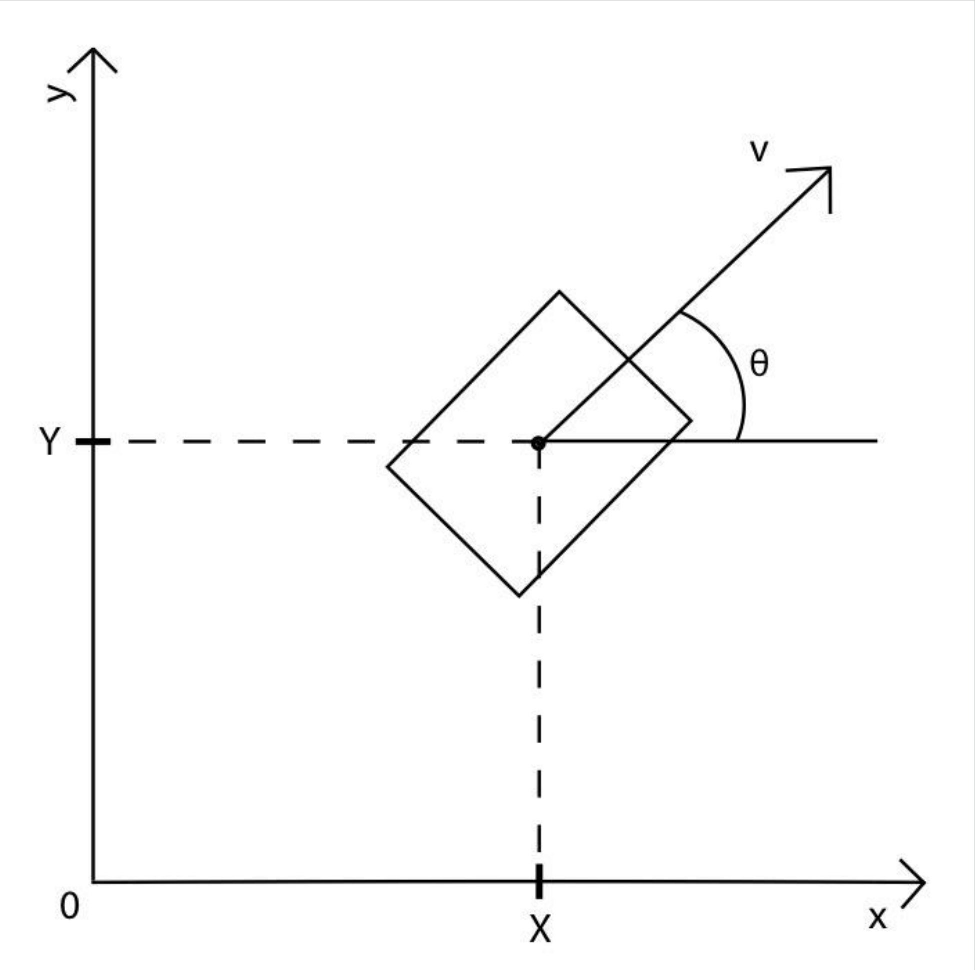


III. QBot 2 – Ryan & Greg

QBot 2 - Ryan

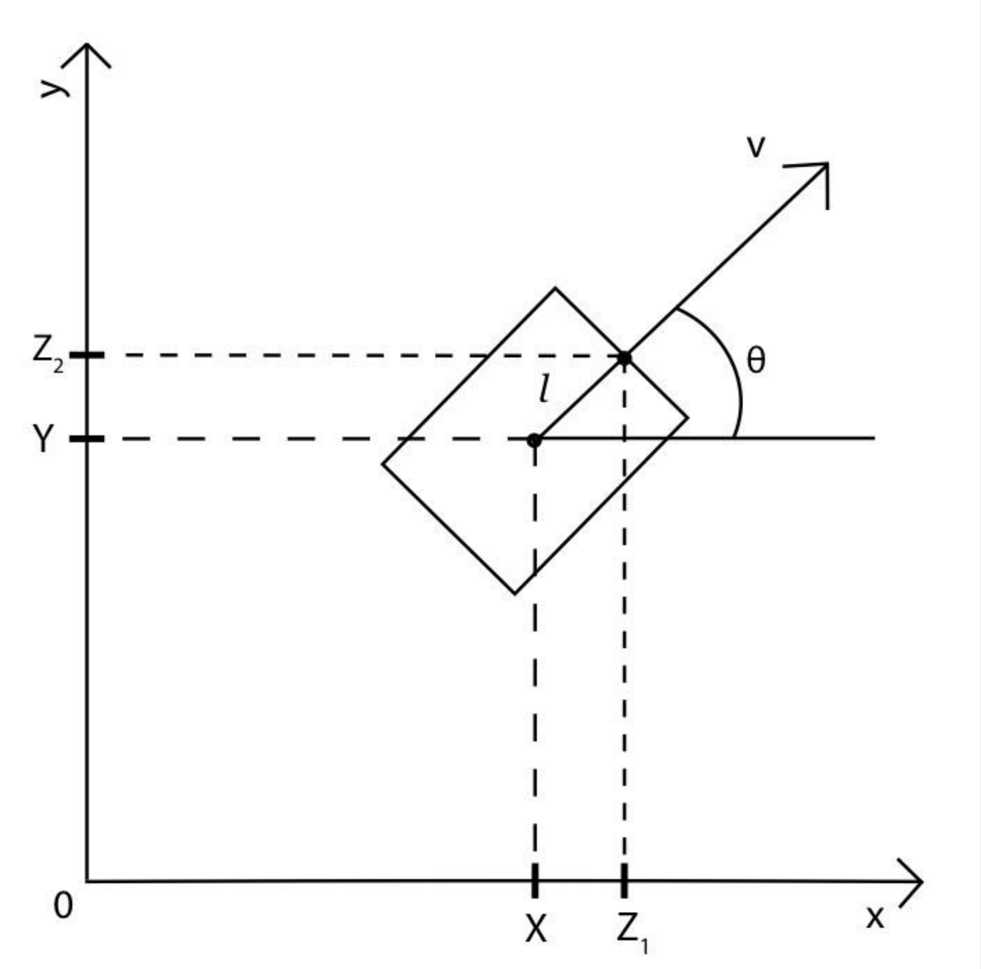
Non-linear Model

- Non-linear Model
  - $\dot{x} = v cos(\theta)$
  - $\dot{y} = v sin(\theta)$
  - $\dot{\theta} = \omega$

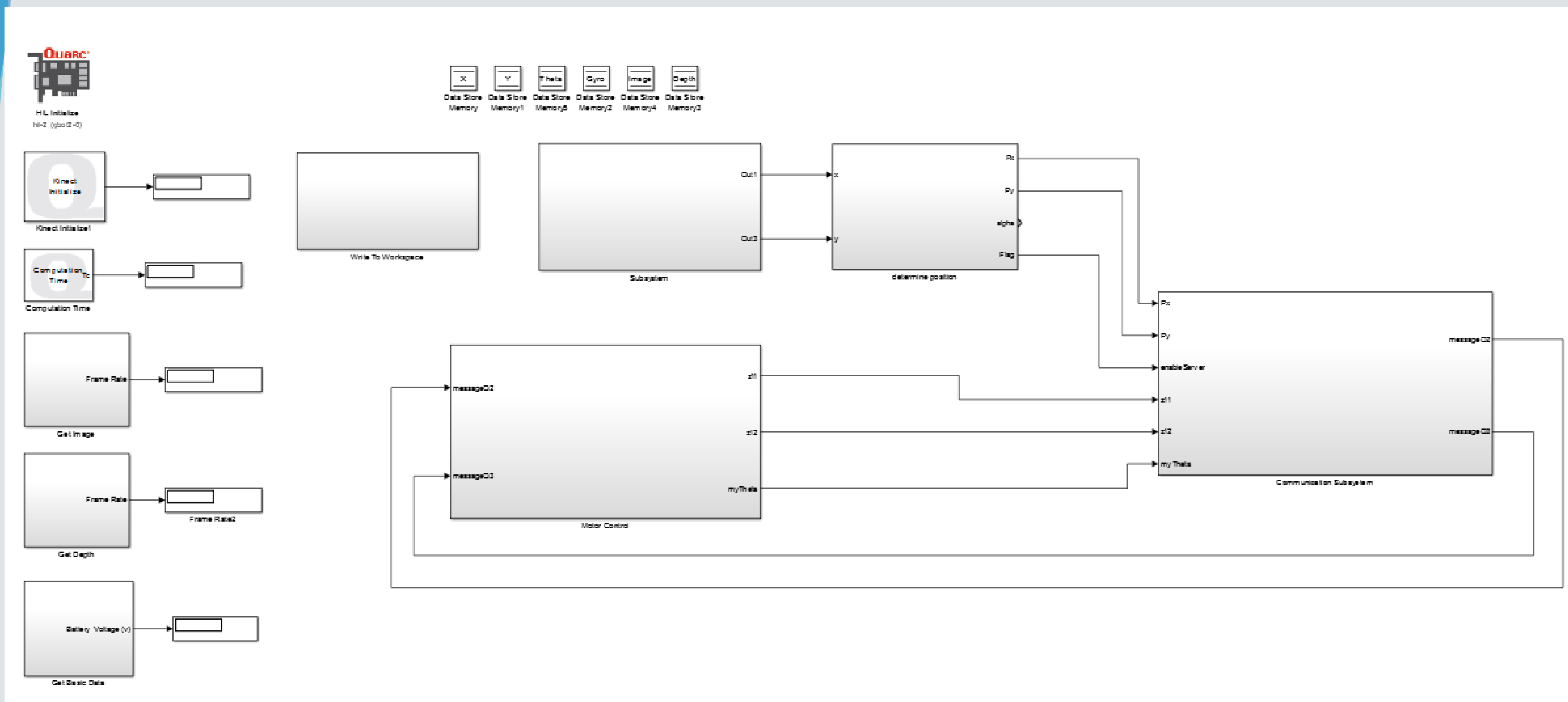


Linear Model

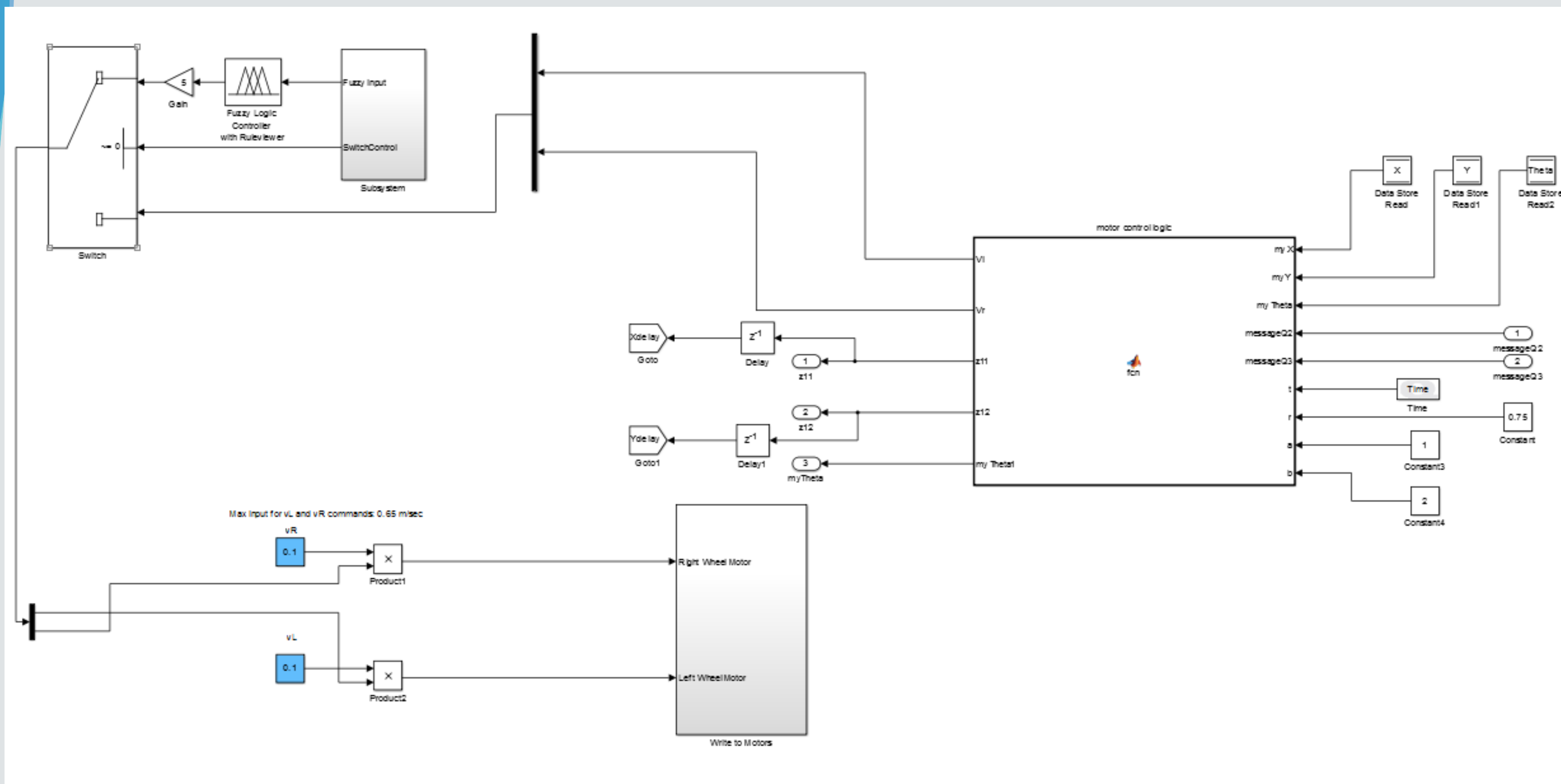
- Linear Model
  - $\dot{p}_x = u_x$
  - $\dot{p}_y = u_y$
  - $p_x = x + l * cos\theta$
  - $p_y = y + l * sin\theta$



Simulink Model

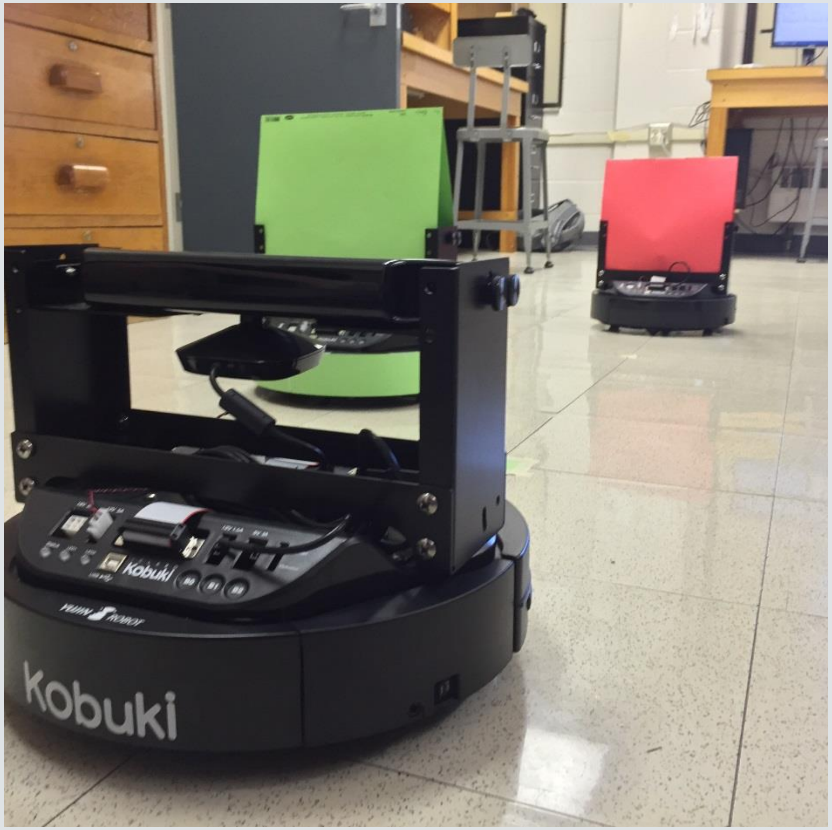


Simulink Model

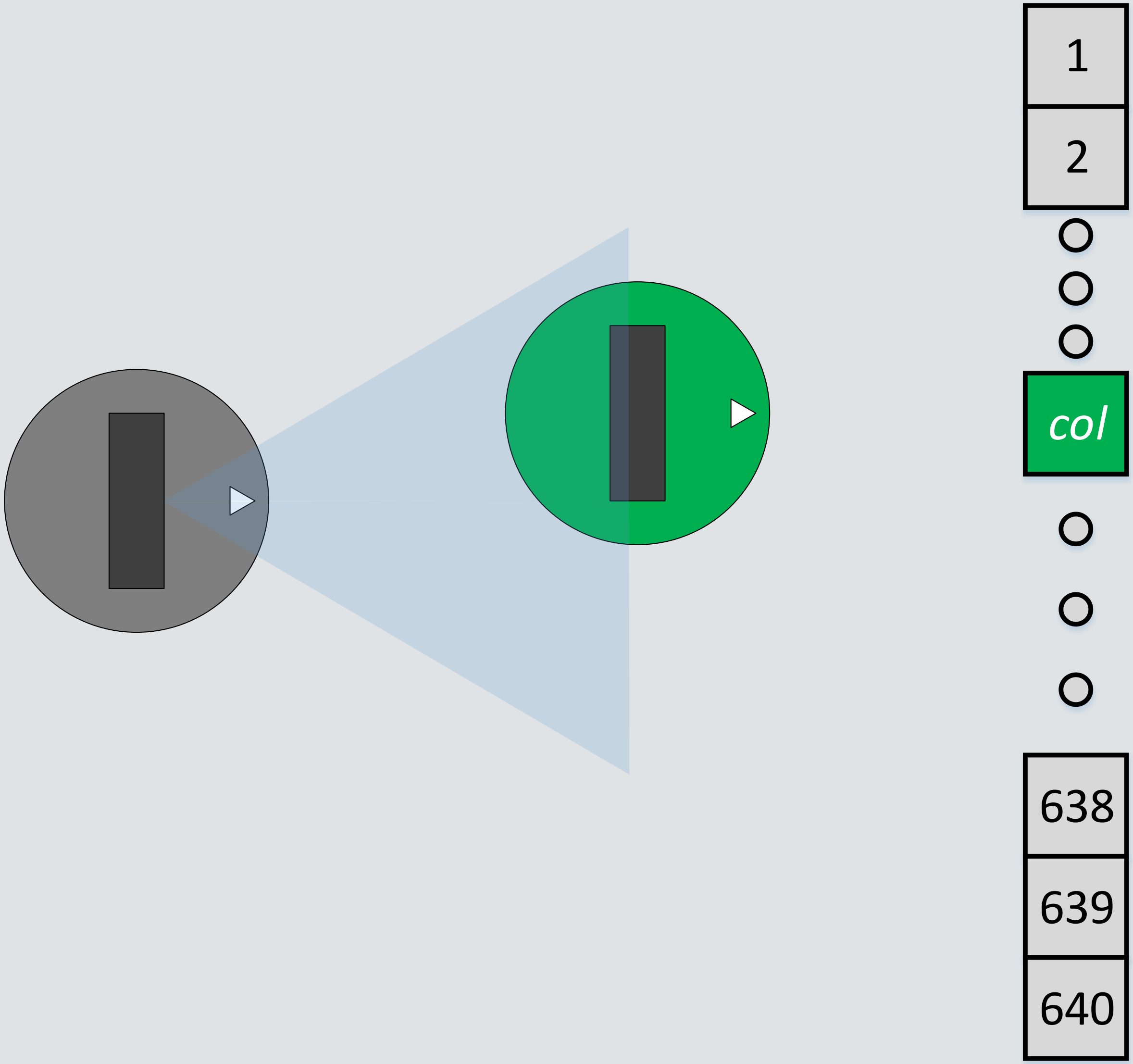


Localization

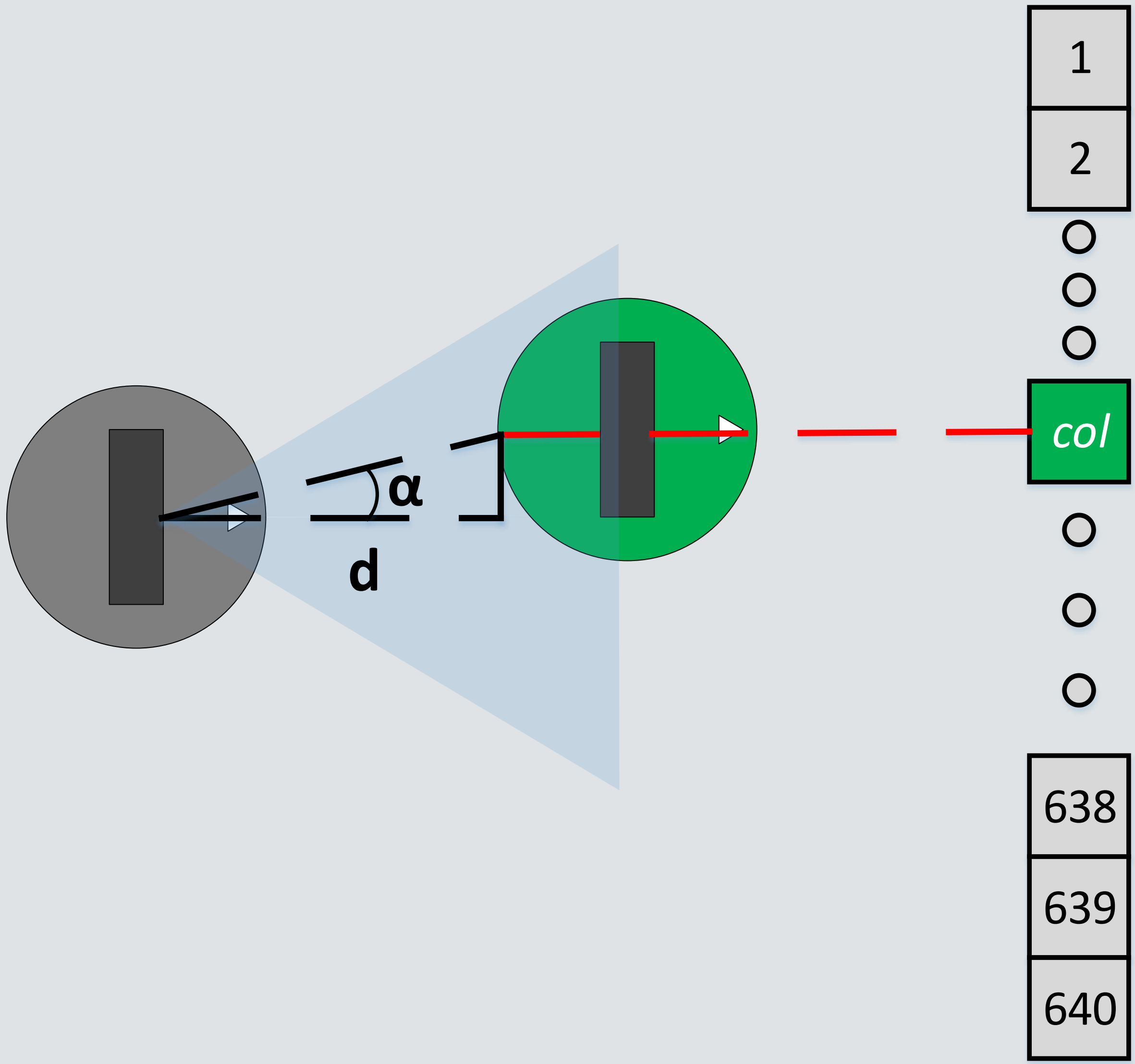
- Color Detection
- Depth Calculation
- Communication



Localization – Color Detection



Localization – Depth Calculation

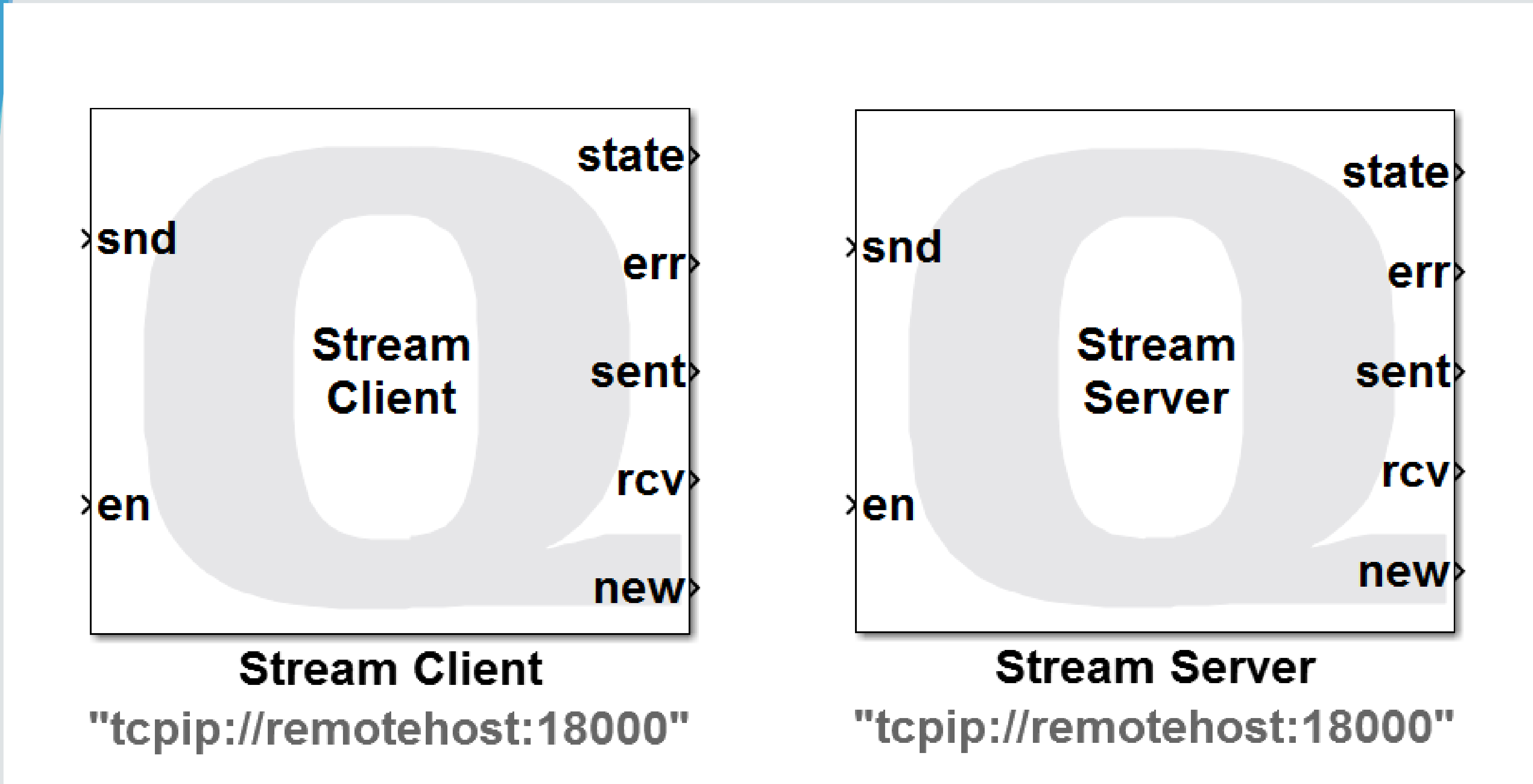


Localization – Depth Calculation

- $\alpha = (320\ - column) * (57/640) * (\pi/180)$
  - α is obtained angle
  - $column$ is the array column number

- $P_x = d$
- $P_y = d * tan(\alpha)$
  - d is depth

Localization – Communication



Point Consensus Control Algorithm

$$u_{ix}(t) = k_i \sum_{j=1}^{n} s_{ij}(t)\left(p_{jx}(t) - p_{ix}(t)\right)$$

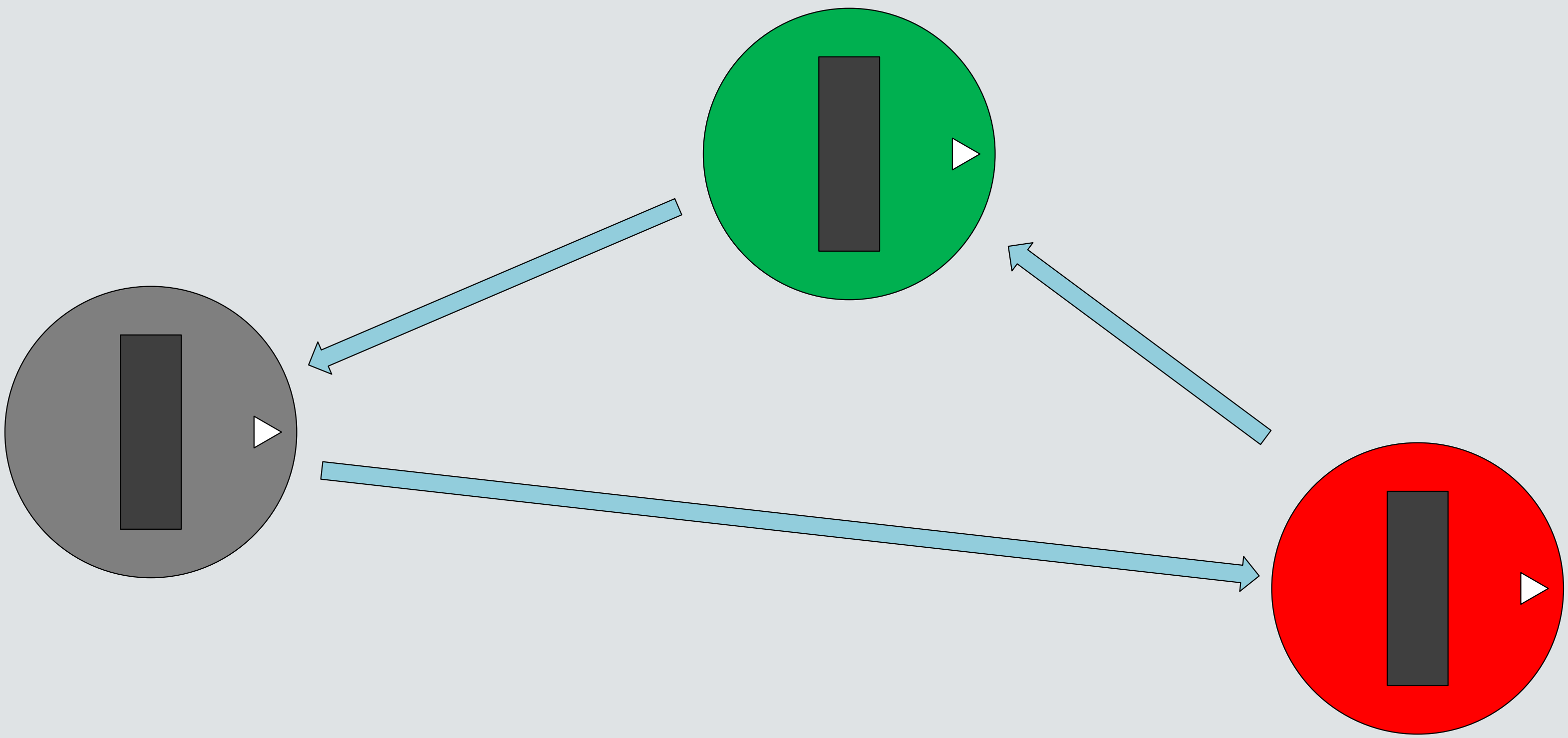
$$u_{iy}(t) = k_i \sum_{j=1}^{n} s_{ij}(t)\left(p_{jy}(t) - p_{iy}(t)\right)$$

- Communication Topology
- $s_{ij}(t) = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$

Point Consensus

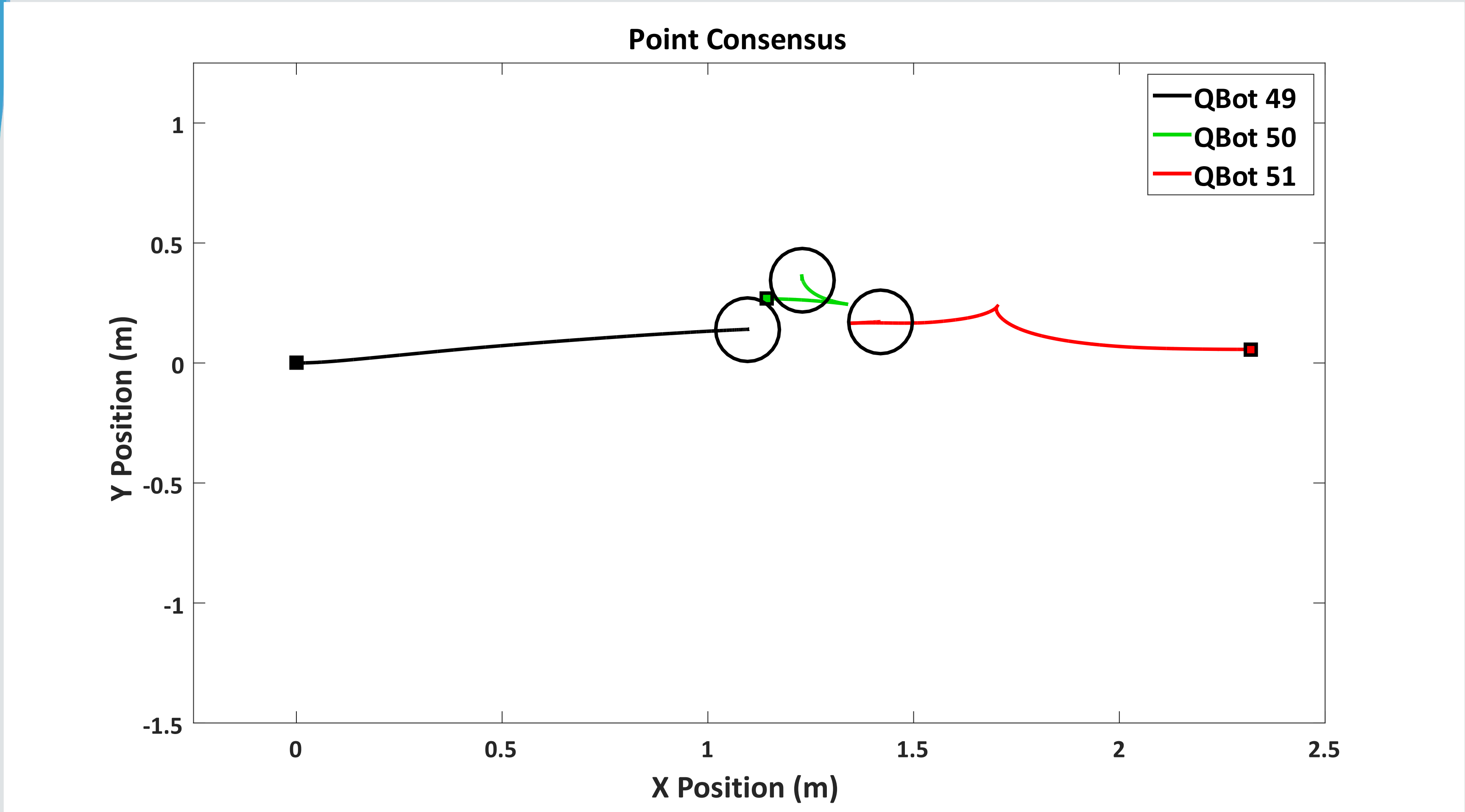
- Communication Topology

$$s_{ij}(t) = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$



Point Consensus

Point Consensus

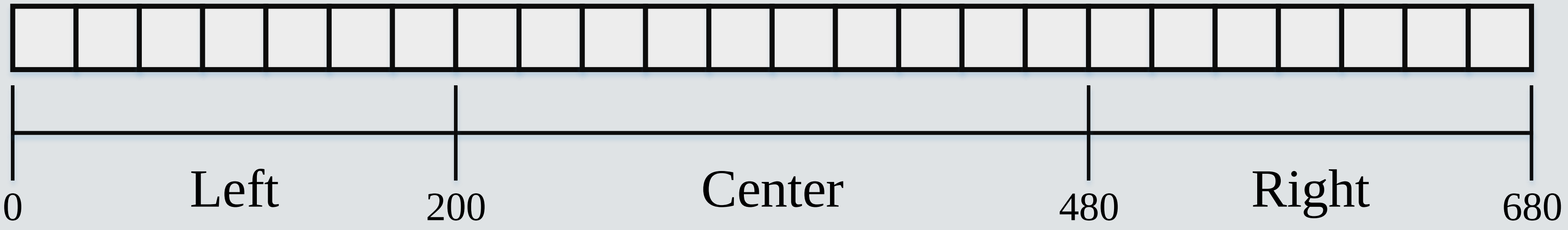


Heading Alignment

QBot 2 - Greg

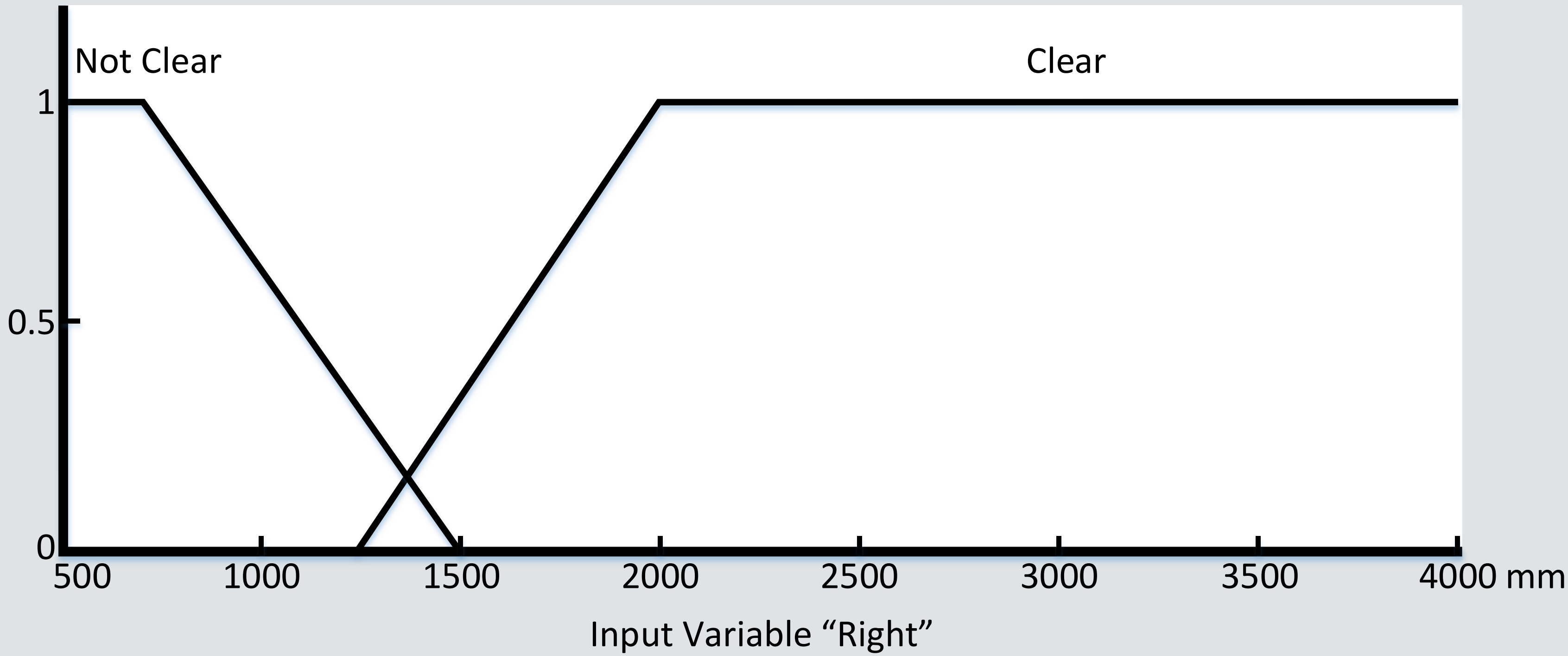
Object Avoidance

- Used Fuzzy Logic
  - Inputs taken from Xbox 360 Kinect
  - Outputs are left and right motor velocities



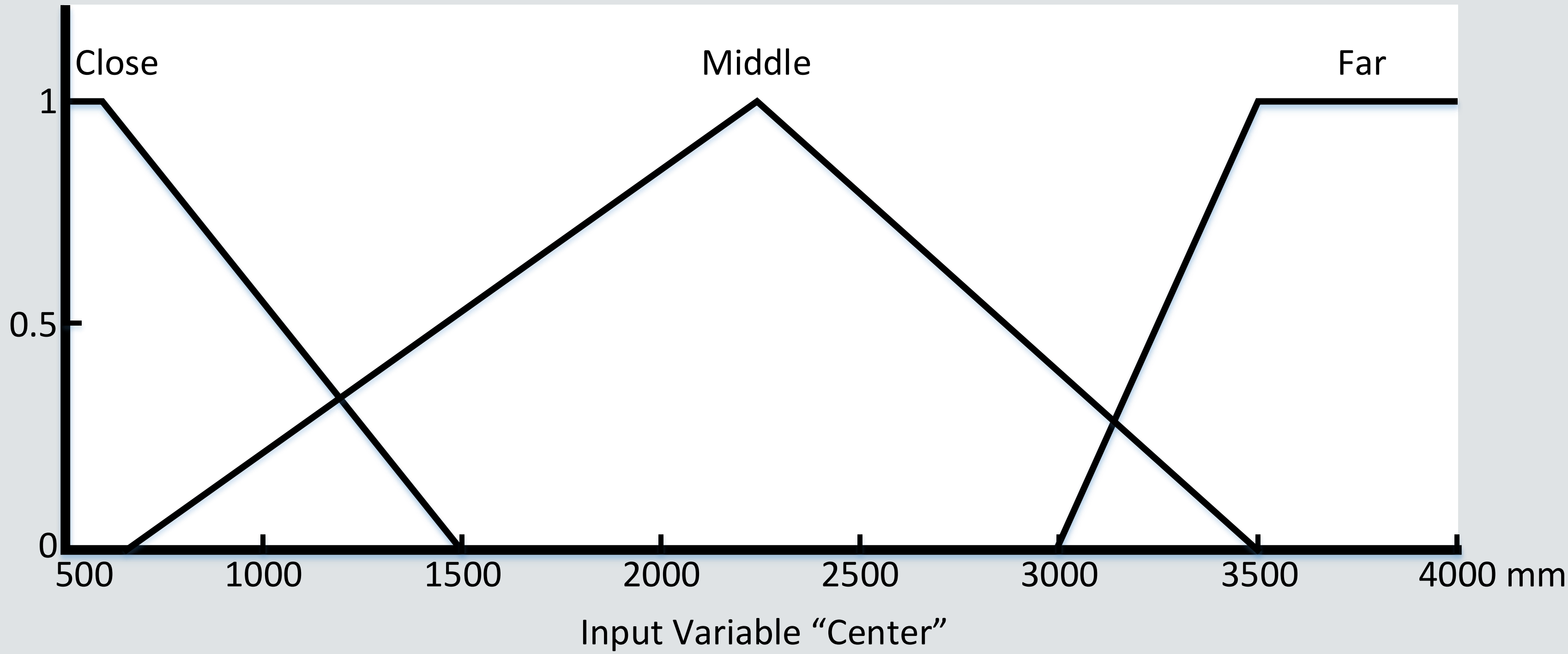
Object Avoidance

Membership Function Plots

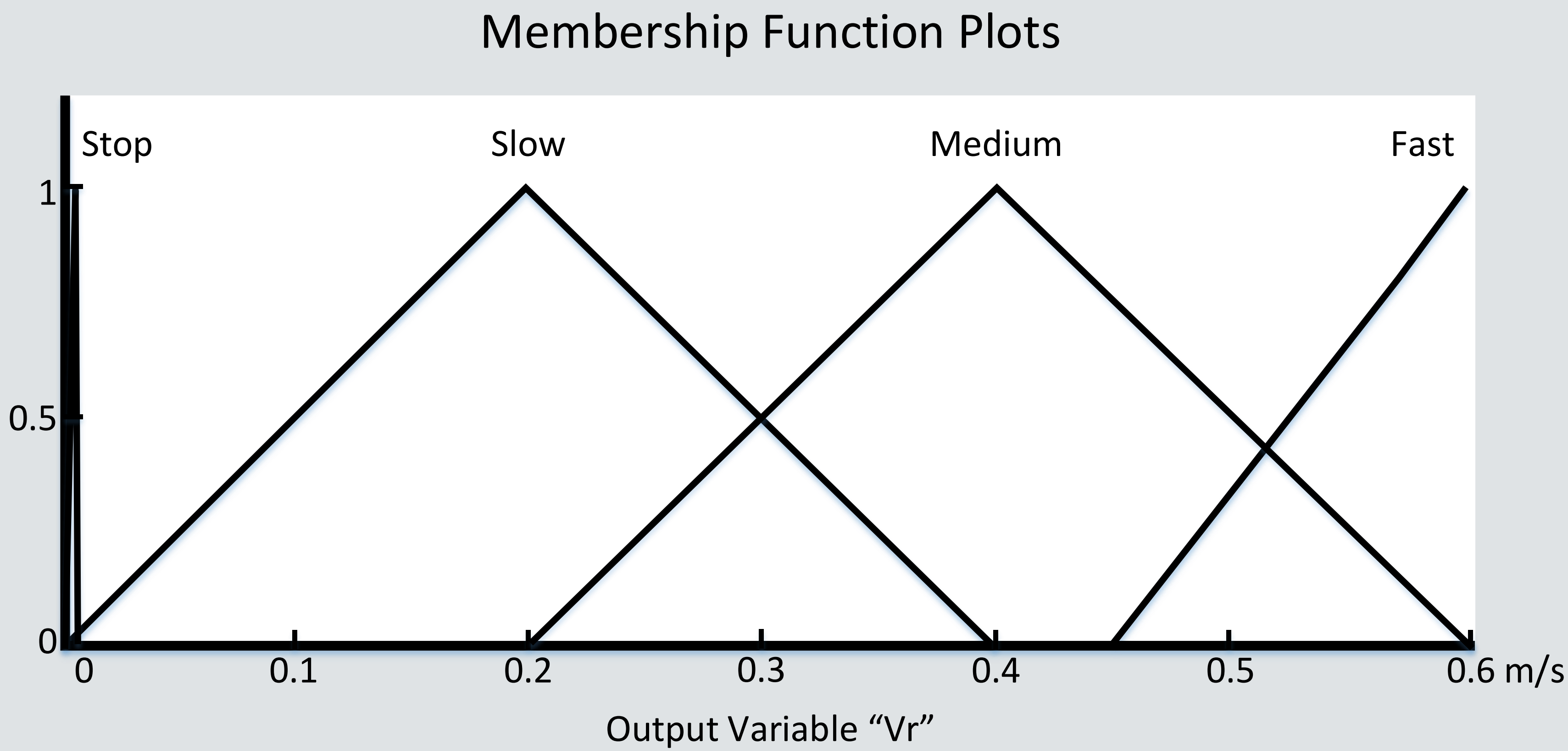


Object Avoidance

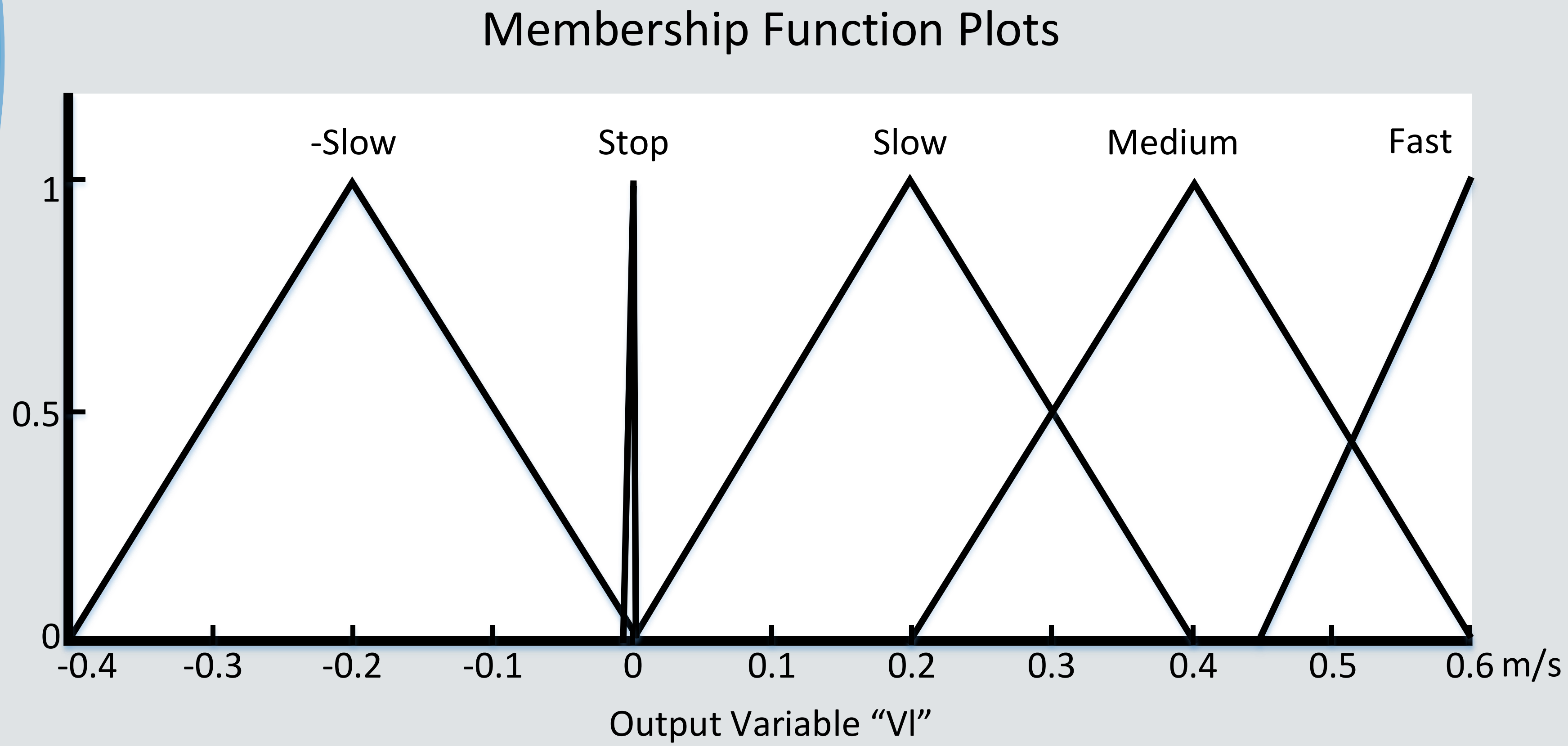
Membership Function Plots



Object Avoidance



Object Avoidance



Fuzzy Rule Set

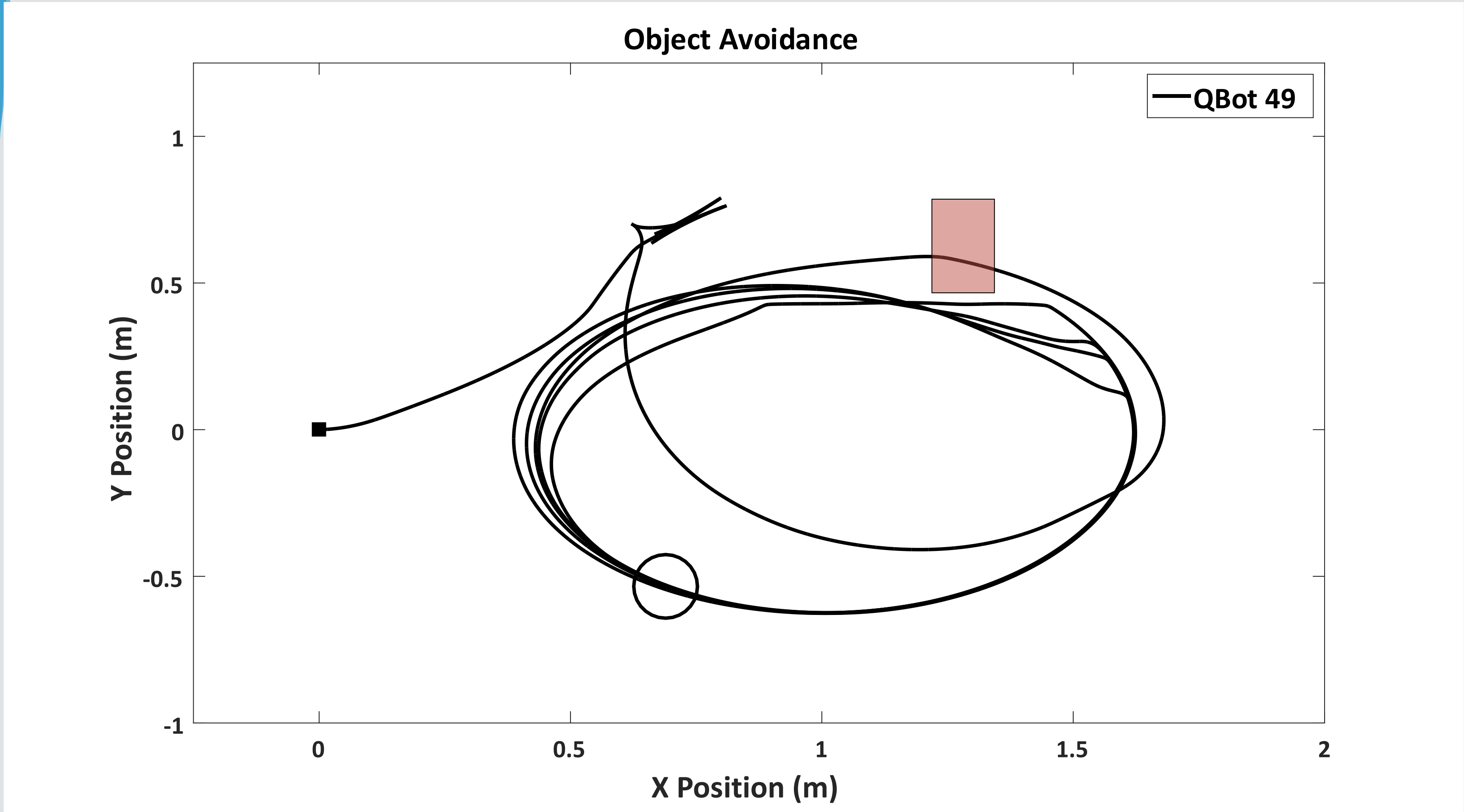
Input			Output	
Left	Center	Right	V_R	V_L
-	Far	-	Medium	Medium
-	Middle	-	Slow	Slow
Not Clear	Close	Not Clear	Slow	-Slow
Clear	Close	Not Clear	Slow	Stop
Not Clear	Close	Clear	Stop	Slow
Clear	Close	Clear	Slow	Stop

Object Avoidance

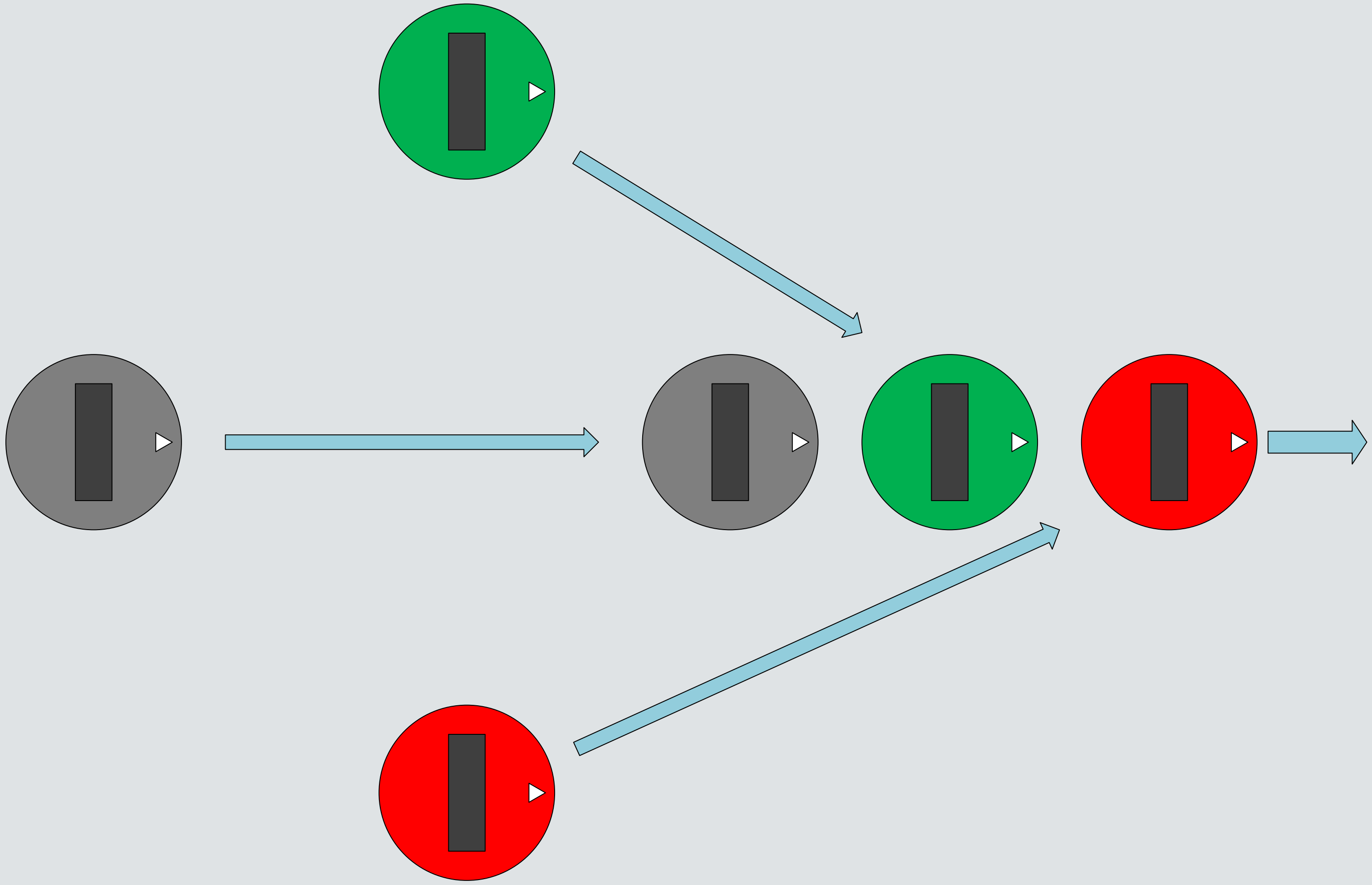
- $V_L = k(x_d - x) + \dot{x}_d + \Delta V_L$
- $V_R = k(y_d - y) + \dot{y}_d + \Delta V_R$

Object Avoidance

Object Avoidance



Formation Control



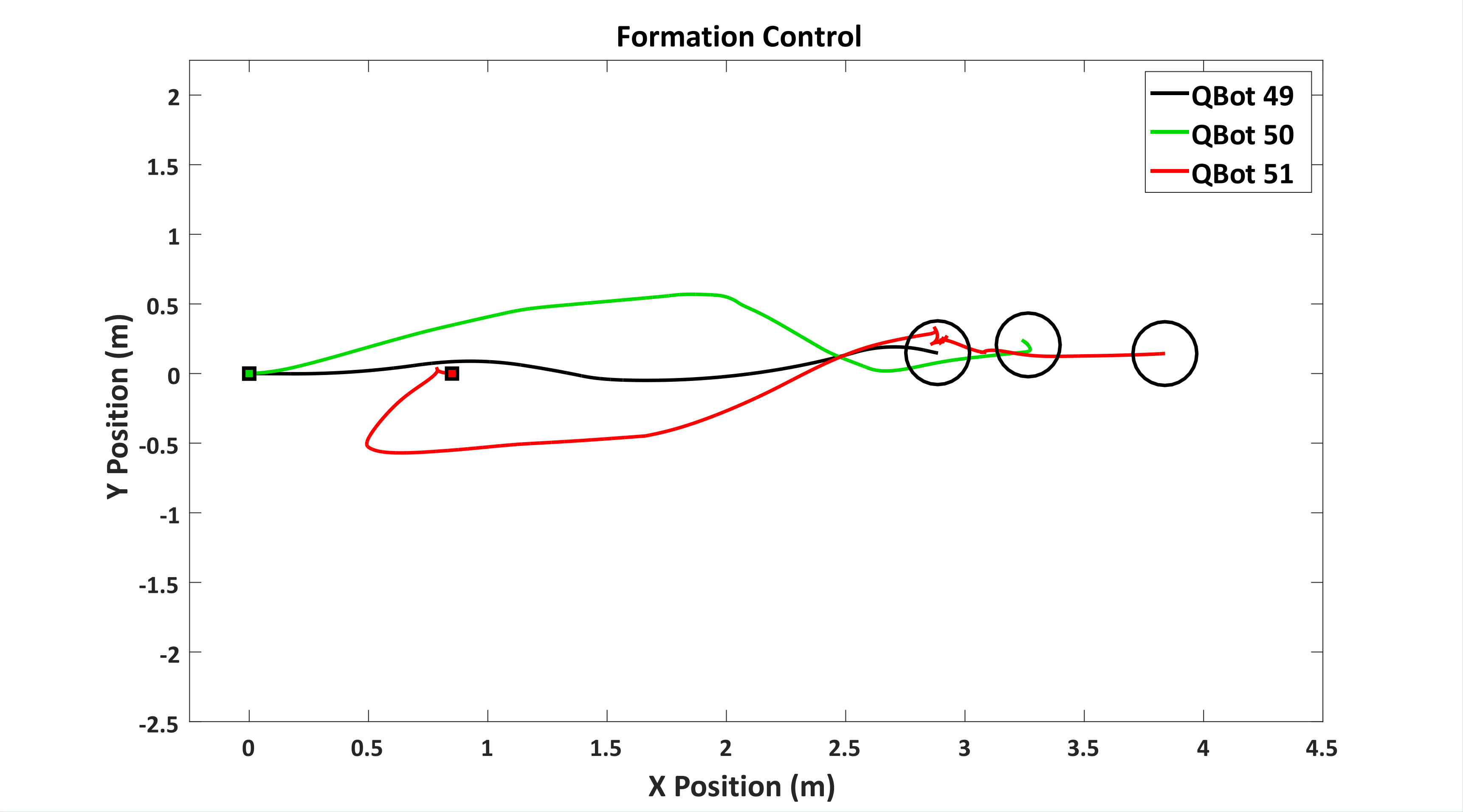
Formation Control

$$u_{ix}(t) = k_i \sum_{j=1}^{n} s_{ij}(t)\left(p_{jx}(t) - C_{jx} - p_{ix}(t) + C_{ix}\right)$$

$$u_{iy}(t) = k_i \sum_{j=1}^{n} s_{ij}(t)\left(p_{jy}(t) - C_{jy} - p_{iy}(t) + C_{iy}\right)$$

Formation Control

Formation Control



IV. Summary & Conclusions

Problems Encountered

- E-puck
  - CMOS camera
  - For communication between different platforms, additional circuity was needed
- Kilobot
  - Kilobot motors need frequent calibration
  - Small size makes it difficult for QBot 2 to detect
  - Lack of sensory information

Summary & Conclusions

- Designed cooperative control algorithms for heterogeneous groups of robots
- Implemented algorithms on different robot platforms

Future Work

- Cross-platform communication
- Implement E-puck camera
- Further development of formation algorithms
- Complete E-puck to Kilobot communication
- Add IR messaging system to QBot 2
- Improve QBot 2 algorithm to avoid objects consistently

Acknowledgements

- Our group would like to thank Dr. Wang & Dr. Ahn for their support throughout the project.
- Our group would also like to thank Mr. Mattus and Mr. Schmidt for their technical support.

COOPERATIVE CONTROL OF HETEROGENEOUS MOBILE ROBOTS NETWORK

Gregory Bock, Brittany Dhall, Ryan Hendrickson, & Jared Lamkin

Project Advisors: Dr. Jing Wang & Dr. In Soo Ahn

Department of Electrical and Computer Engineering

April 26th, 2016

Division of Labor Overview

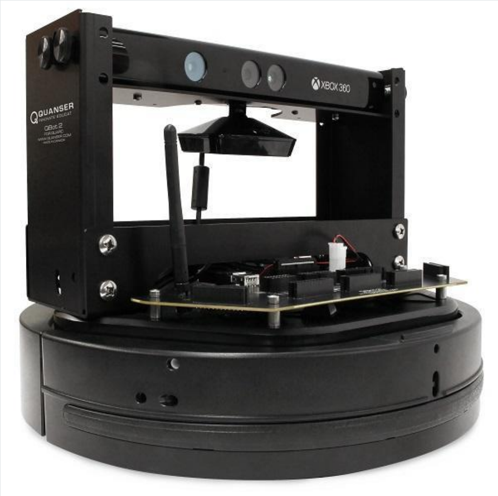
Individual Behavior	Kilobots	Jared
	QBot 2s	Ryan/Greg
	E-pucks	Brittany/Jared
Individual Communication	Kilobot - Kilobot	Jared
	QBot - QBot	Ryan/Greg
	E-puck - E-puck	Brittany/Jared
Integrated Communication	Kilobot - E-puck	Jared/Brittany
	Kilobot - QBot	Jared/Ryan/Greg
	E-puck - QBot	Brittany/Ryan/Greg
Algorithm Design	Linearization Based Model	Jared/Brittany/Ryan/Greg
Integrated Behavior	Formation Control Behavior	Jared/Brittany/Ryan/Greg
	Flocking Behavior	Jared/Brittany/Ryan/Greg
Testing	Software Implementation	Jared/Brittany/Ryan/Greg
	Hardware Implementation	Jared/Brittany/Ryan/Greg

Algorithm Test Platforms

Kilobot

E-Puck

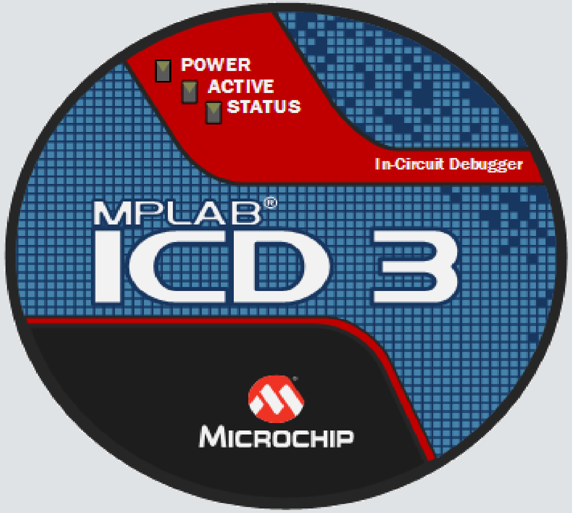
QBot 2



http://www.k-team.com/mobile-robotics-products/kilobot
https://en.wikipedia.org/wiki/E-puck_mobile_robot
http://www.mathworks.com/products/connections/product_detail/product_101072.html

Unbricking the E-pucks

- Uses the MPLAB ICD 3 In-circuit Debugger
- MPLAB IDE v8.30
- Erases the Flash memory by powering the E-puck through the ICD 3



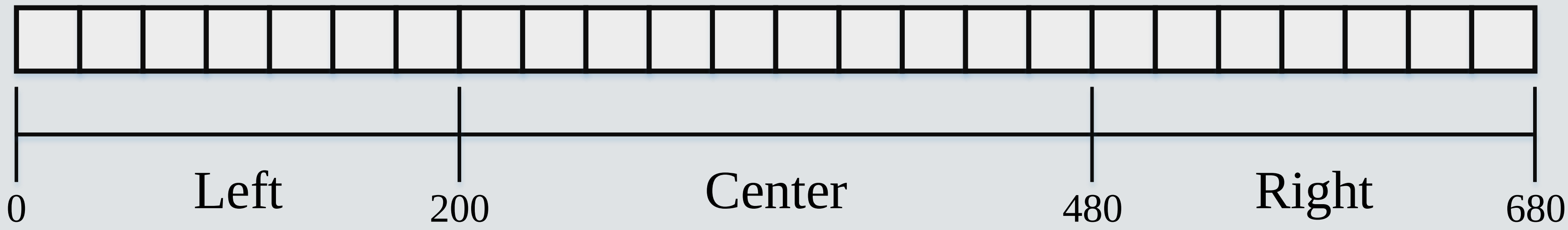
http://microchip.wikidot.com/icd3:start

Changing the Original Timer

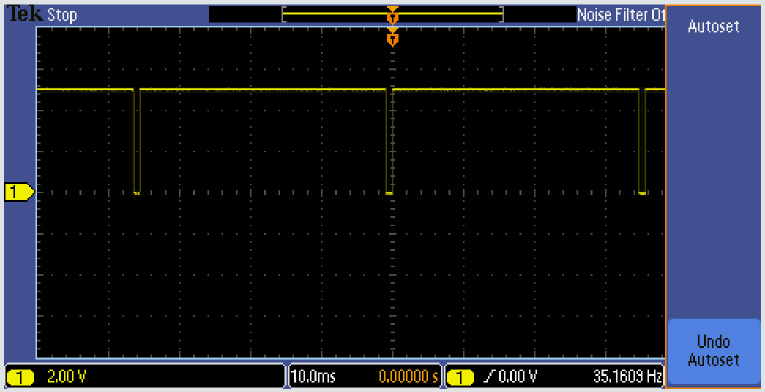
- E-puck's clock speed is 8 times faster than the Kilobot
- Increased the timer of the E-puck by a factor of 8 to slow down the rate at which the message was sent to Kilobot
- Change was made to allow Kilobots to sync with E-puck messaging

Object Avoidance

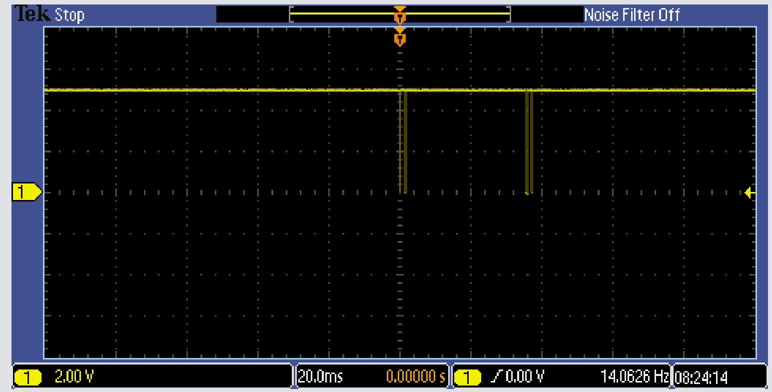
- Inputs taken from Xbox 360 Kinect



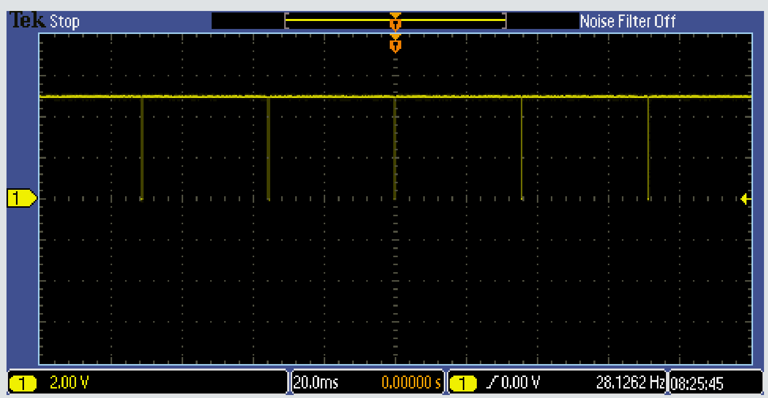
Oscilloscope Screen Captures from the Infrared Receiver Circuit



Original timer used in initial Kilobot testing



Increased original timer by a factor of 8



Decreased original timer by a factor of 8

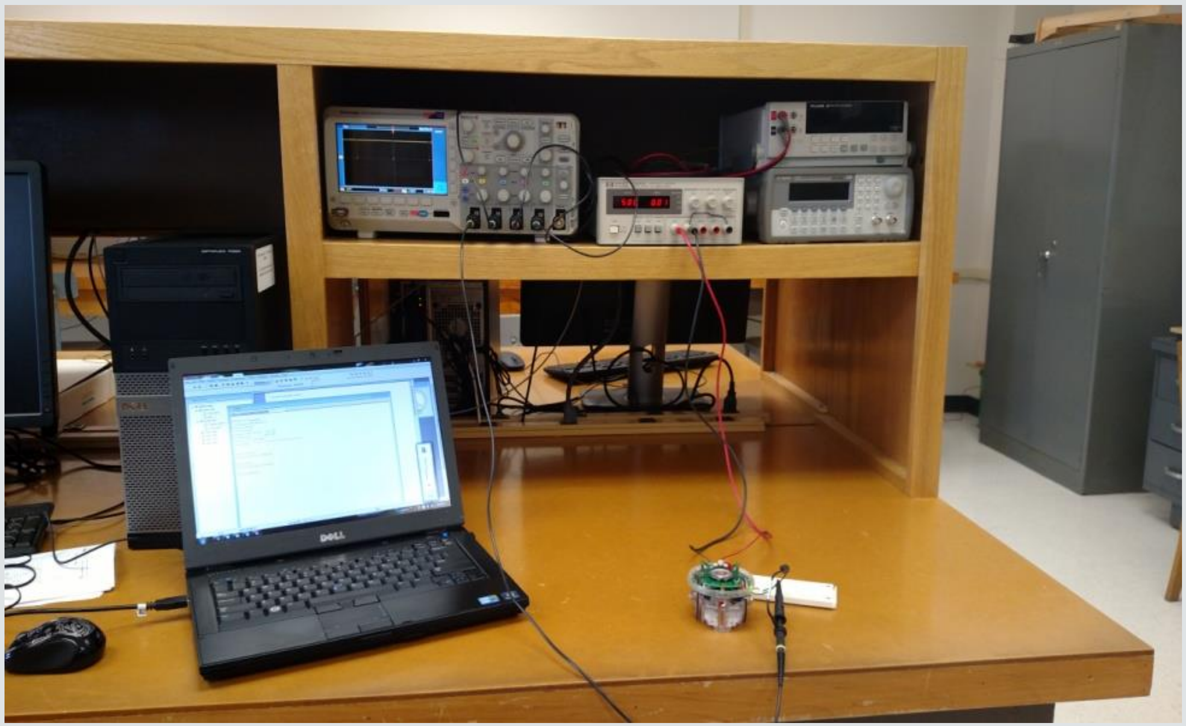
Advanced Localization: Distributed Trilateration

- Gradient is only a 1D localization
- Minimum of 3 fixed agents as reference points
- Non-localized agents assume position (0,0)
- Determine actual distance to non-localized agent
- Calculate assumed distance

Advanced Localization: Distributed Trilateration

- Direction Vectors are generated from reference to unknown
- Generate assumed coordinates from Vectors and measured Distances
- New position is determined using assumed position and previous position

Integrated Communication Set-up



Project Platform Costs

Platform	Quantity	Total Price
QBot 2	3	$9,999.00
Kilobot Kit	20	$4,583.00
Epucks	3	$5,093.00

Programming Software Costs

Software	Quantity	Total Price
Kilobot Controller IDE	1	$0.00
E-puck Programming Software	1	$0.00
MATLAB Courseware	1	$0.00

E-puck Object Following Code

```
/**********************************************************************/
/** Motor speed controlled depending on front proximity sensor values **/
/**********************************************************************/

#include "p30f6014A.h"
#include "e_epuck_ports.h"
#include "e_init_port.h"
#include "e_ad_conv.h"
#include "e_prox.h"
#include "e_motors.h"

#define DELAY 50000

int main() {

    long timer = 0;

    //system initialization
    e_init_port();              // configure port pins
    e_init_ad_scan(ALL_ADC);    // configure Analog-to-Digital Converter Module

    while (1) {

        if (e_get_prox(0)> 500) {       //escape
            e_set_speed_left(0);
            e_set_speed_right(0);
        } else if (e_get_prox(0)>100) { //follow
            e_set_speed_left(400);
            e_set_speed_right(400);
        } else {                        //stop
            e_set_speed_left(0);
            e_set_speed_right(0);
        }

        //wait a little to let the robot move
        for(timer = 0; timer < DELAY; timer++);

    }
}
```

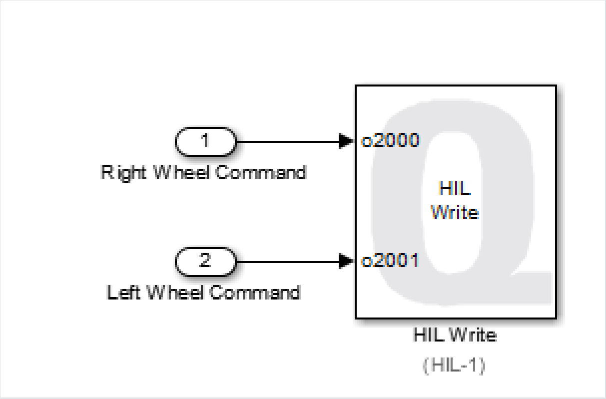
QBot Point Convergence Code

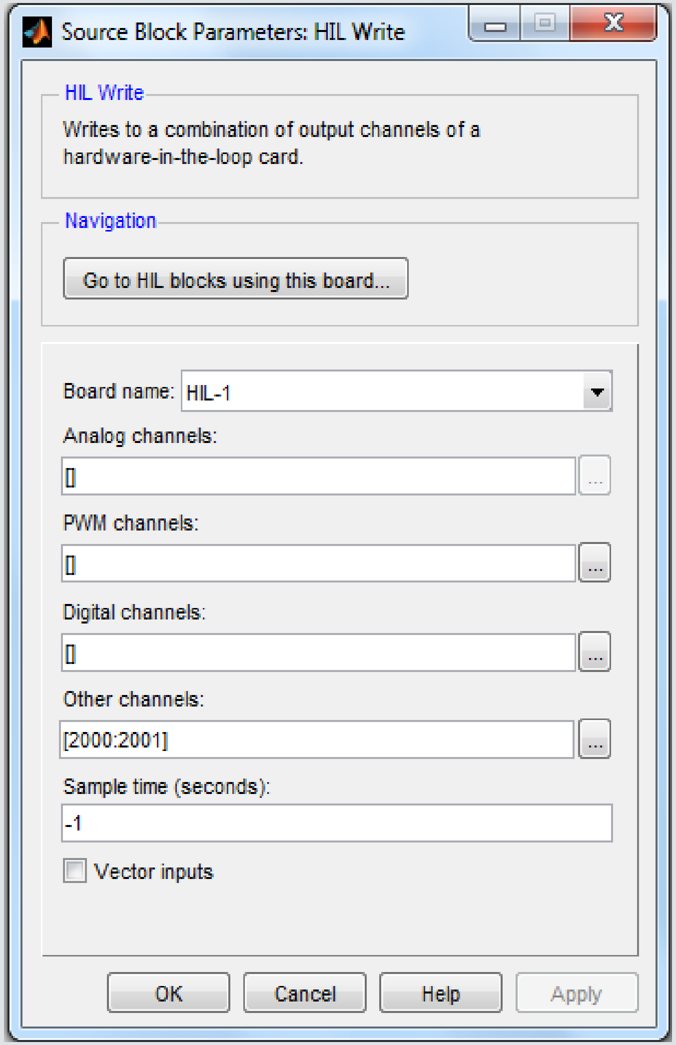
```
v11 = k1*(z21 - z11); % Calculate velocity in x direction
v12 = k2*(z22 - z12); % Calculate velocity in y direction
mat = [cos(myTheta) -d/2*sin(myTheta); sin(myTheta) d/2*cos(myTheta)];
myControl = inv(mat)*[v11;v12];

% Determine total velocity
V = myControl(1);

% Determine angular velocity
omega = myControl(2);

% Determine left and right wheel velocity
Vl = (2*V-d*omega)/2;
Vr = (2*V+d*omega)/2;
```

QBot Obtained Angle Equation

- $\alpha = (320 - column) * (57/640) * (\pi/180)$

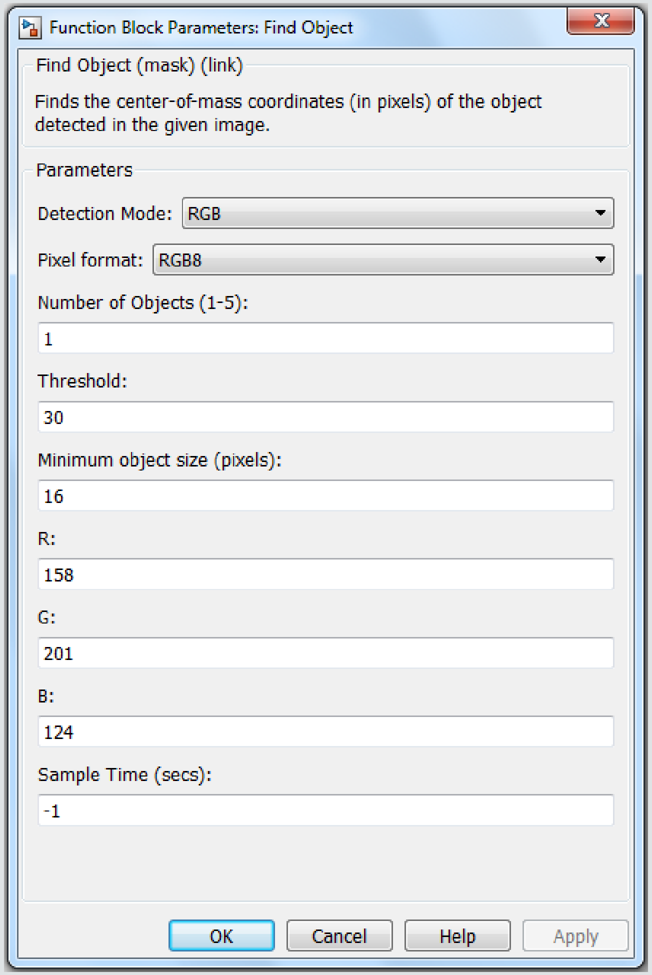
HIL Write Block



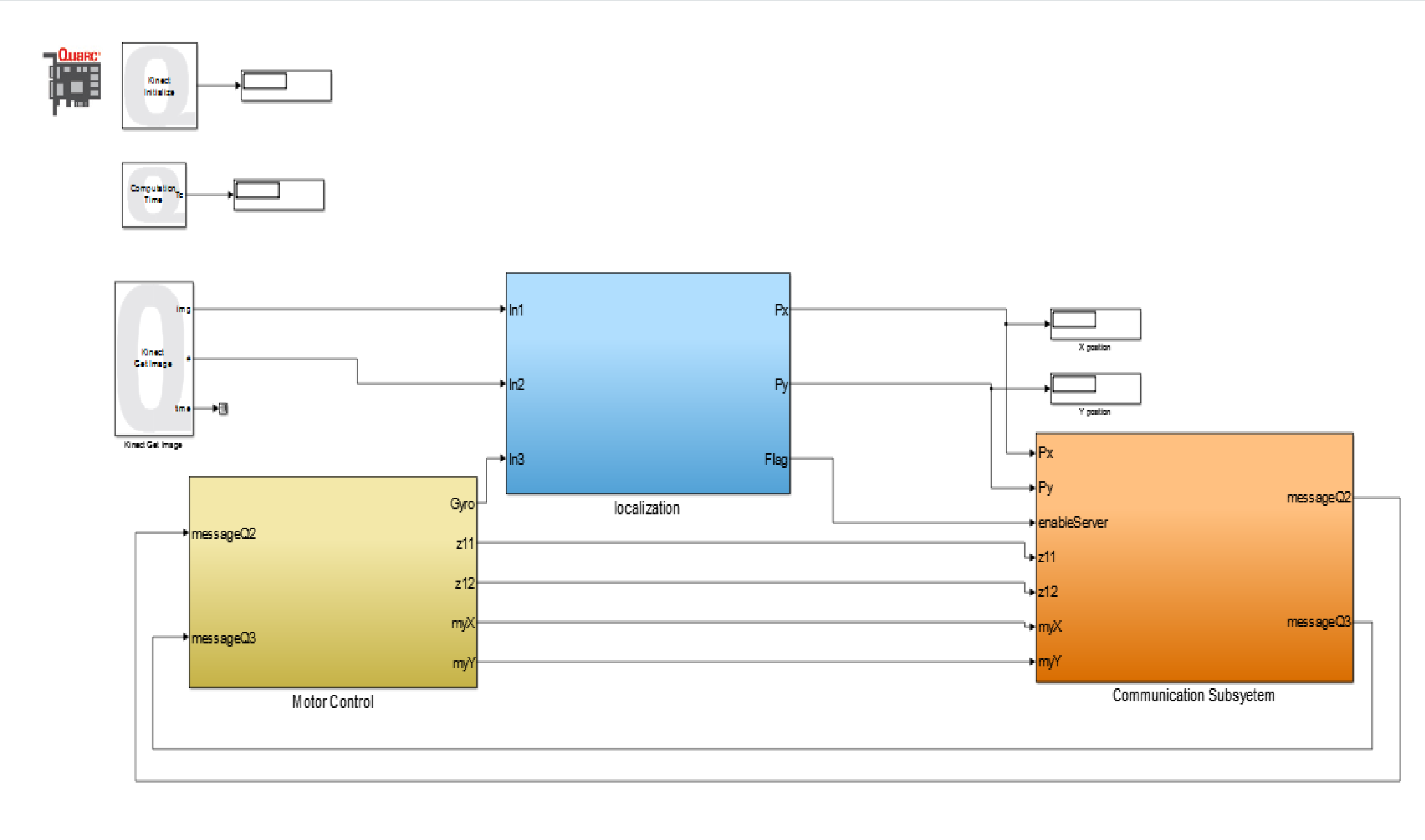


Find Object Parameters

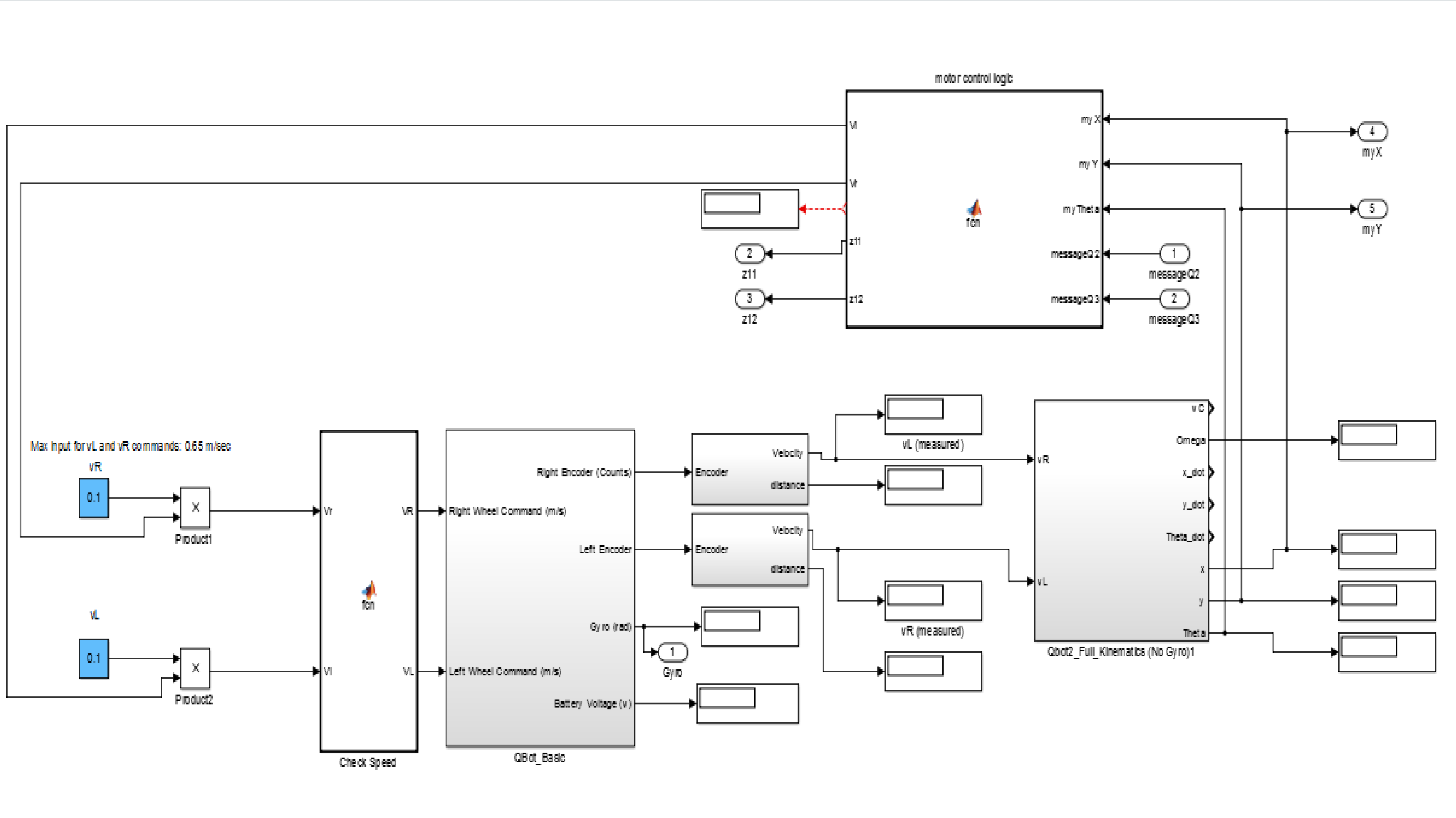
- Specify RGB values
- Value threshold
- Number of objects



Overall Simulink Model



Motor Control



Localization Equations

- $C_i = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2}$
- $V_i = < \frac{x_0 - x_i}{C_i}, \frac{y_0 - y_i}{C_i} >$
- $n_i = (x_i, y_i) - D_i * V_i$
- $(x_0, y_0) = (x_0, y_0) - \frac{(x_0 - n_{ix}, y_0 - n_{iy})}{4}$

Color Consensus

- Kilobots are initialized with a random number
- Each number corresponds to a color
- Kilobots then begin transmitting value
- Kilobots receive messages and keep track of how many neighbors are what color
- Kilobots then change their color to most prevalent color

Color and Object Detection

- The E-puck CMOS camera is capable of 640X480 resolution, in color or grayscale
- However, the image is too large to process, so instead we use a 1X120 image
- Color uses RGB565, where each pixel has 5 bits for red, 6 bits for green, and 5 bits for blue

Color and Object Detection

- First step to object detection is edge detection
- The image array is searched for two edges, from both left and right starting positions
- Individual pixels are compared to the average of the previous ten pixels
- If the difference is greater than three, that location is set as an edge
- Based on the number of edges found (0,1,2,3,4), The E-puck calculates where the center of the object is, and how wide it is.

Color and Object Detection

- After Edge detection is complete, the E-puck moves on to color comparison
- The E-puck computes the average RGB value of the object
- The average is compared to the specified value within a certain tolerance
- If the comparison is acceptable, The E-puck begins maneuvering to it.

Odometry

- $\triangle\ \theta = \frac{(\triangle R - \triangle L)}{2}$
- $\triangle\ S = \frac{(\triangle R + \triangle L)}{2}$
- $\triangle\ x = \triangle\, S * cos\left(\theta + \frac{(\triangle\theta)}{2}\right)$
- $\triangle\ y = \triangle\, S * sin\left(\theta + \frac{(\triangle\theta)}{2}\right)$
- $x(k+1) = x(k) + \triangle\, x$
- $y(k+1) = y(k) + \triangle\, y$
- $\theta(k+1) = \theta + \frac{(\triangle\theta)}{3}$