Advanced Mechatronics:
Final Project

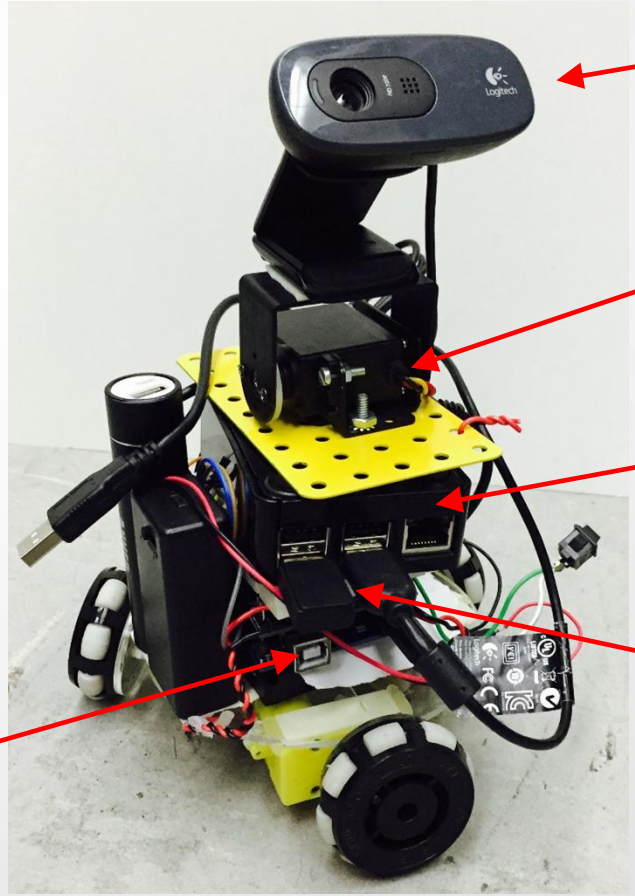
Ball tracking with Omni-Directional Robot

Anthony Brill, Matthew Moorhead, Jonghyun Bae

Ball-tracking Omni-bot Intro

- Recognizing Raspberry Pi's powerful capability of vision processing, we added more functionality to our omni-directional robot, where the robot is able to distinguish different colors of objects and tracks the user-selected ball
- For this project, we have configured all the processes on a mobile platform
- Raspberry Pi is performing real-time color segmentation to distinguish the user-selected object from other objects and identifies the position of the object
- Arduino receives the position of the object, and gives commands to the motors to align the robot with the center of the object

Hardware Components



Logitech c270

Hitec
HS-645MG

Raspberry Pi 2

DK Electronics
Motor Shield

Arduino Uno

Omni-Directional Robot

Theory of Operation

$$v = v_a + v_b + v_c \quad (1)$$

$$v_a = -\omega \bar{a}_3 \times r\bar{a}_2 = \omega_A r\bar{a}_1 = -\omega_A r\bar{x} = -v_x \quad (2)$$

$$v_b = \omega_B \bar{b}_3 \times r\bar{b}_2 = -\omega_B r\bar{b}_1 = \omega_B r\left(\tfrac{1}{2}\bar{x} + \tfrac{\sqrt{3}}{2}\bar{y}\right) = \tfrac{1}{2}v_x + \tfrac{\sqrt{3}}{2}v_y \quad (3)$$

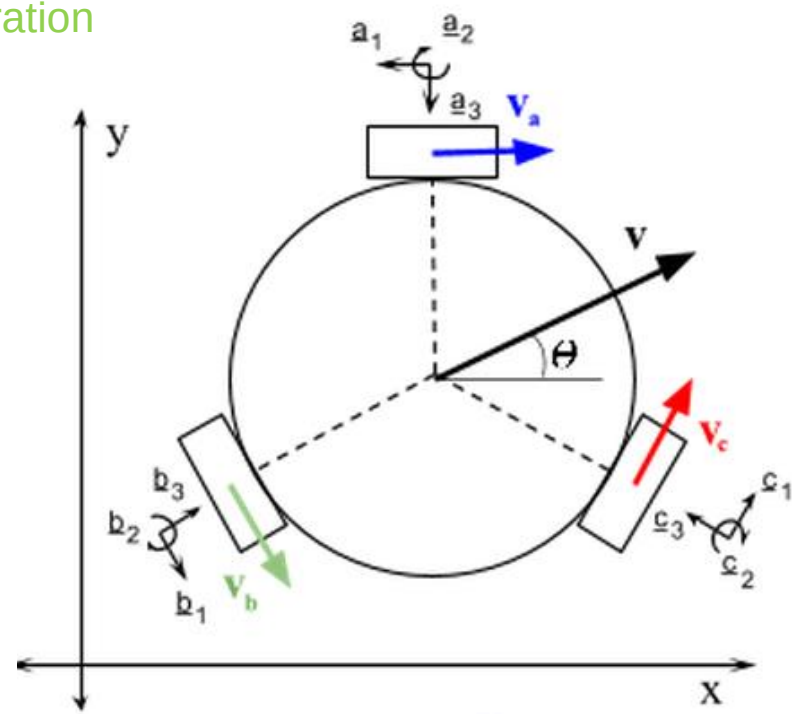
$$v_c = \omega_C \bar{c}_3 \times r\bar{c}_2 = -\omega_C r\bar{c}_1 = \omega_C r\left(\tfrac{1}{2}\bar{x} - \tfrac{\sqrt{3}}{2}\bar{y}\right) = \tfrac{1}{2}v_x - \tfrac{\sqrt{3}}{2}v_y \quad (4)$$

$$v_x = \|v\| \cos\Theta \quad (5)$$

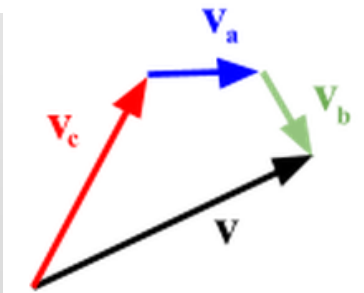
$$v_y = \|v\| \sin\Theta \quad (6)$$

$$\Theta = \tan^{-1}\left(\tfrac{y}{x}\right) \quad (7)$$

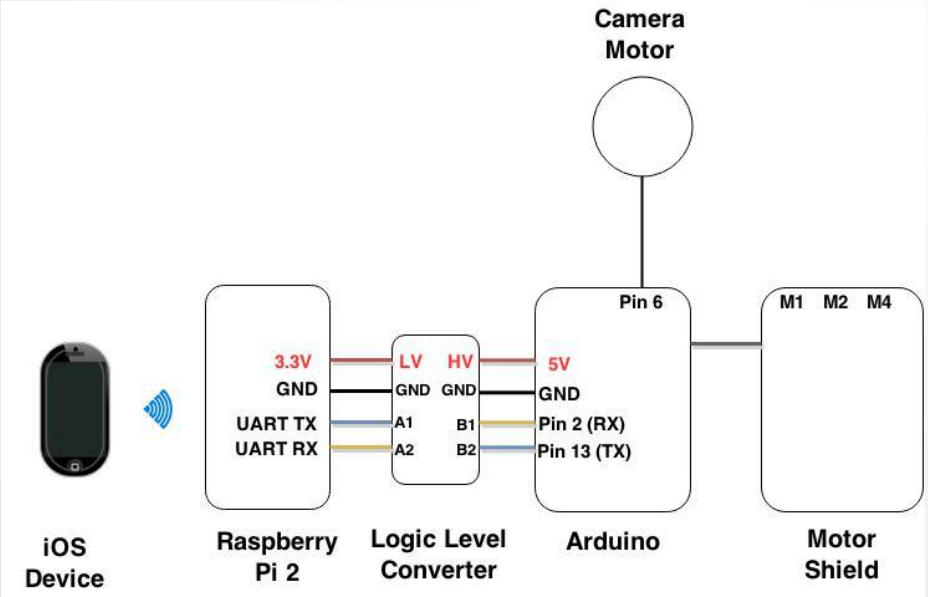
$$\|v\| = \sqrt{x^2 + y^2} \quad (8)$$



	a1	a2	a3	b1	b2	b3	c1	c2	c3
x	-1	0	0	$\frac{1}{2}$	0	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	0	$-\frac{\sqrt{3}}{2}$
y	0	0	-1	$-\frac{\sqrt{3}}{2}$	0	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	0	$\frac{1}{2}$
z	0	-1	0	0	-1	0	0	-1	0

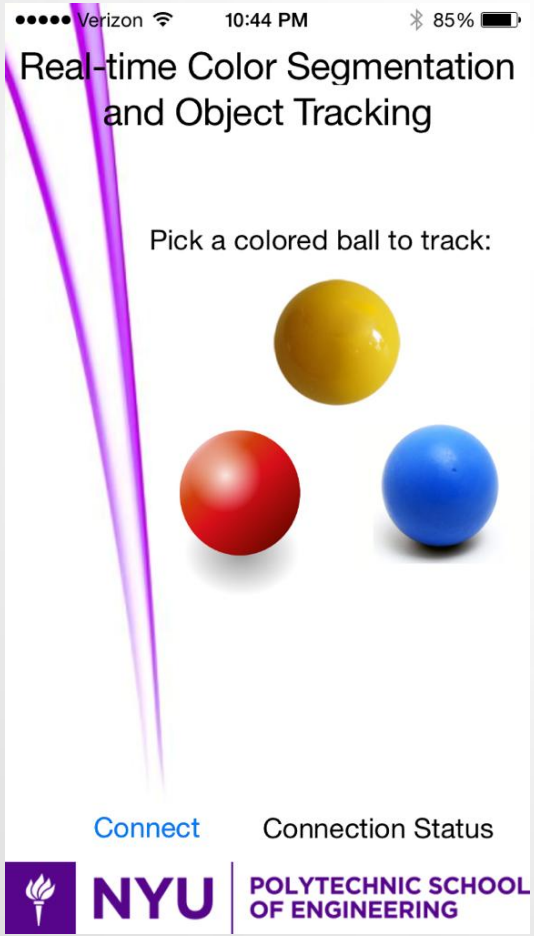


Connection/Communication



- Raspberry Pi is connected to Arduino via Logic Level Converter
  - USART Communication (RX/TX)
  - Softserial is utilized in Arduino (Pin2: RX / Pin13:TX)
- iOS device to RPi: (TTL Serial Comm)
- One additional motor is used to control the pitch of the camera
- Motor shield used to control DC motors

iPhone: User Interface



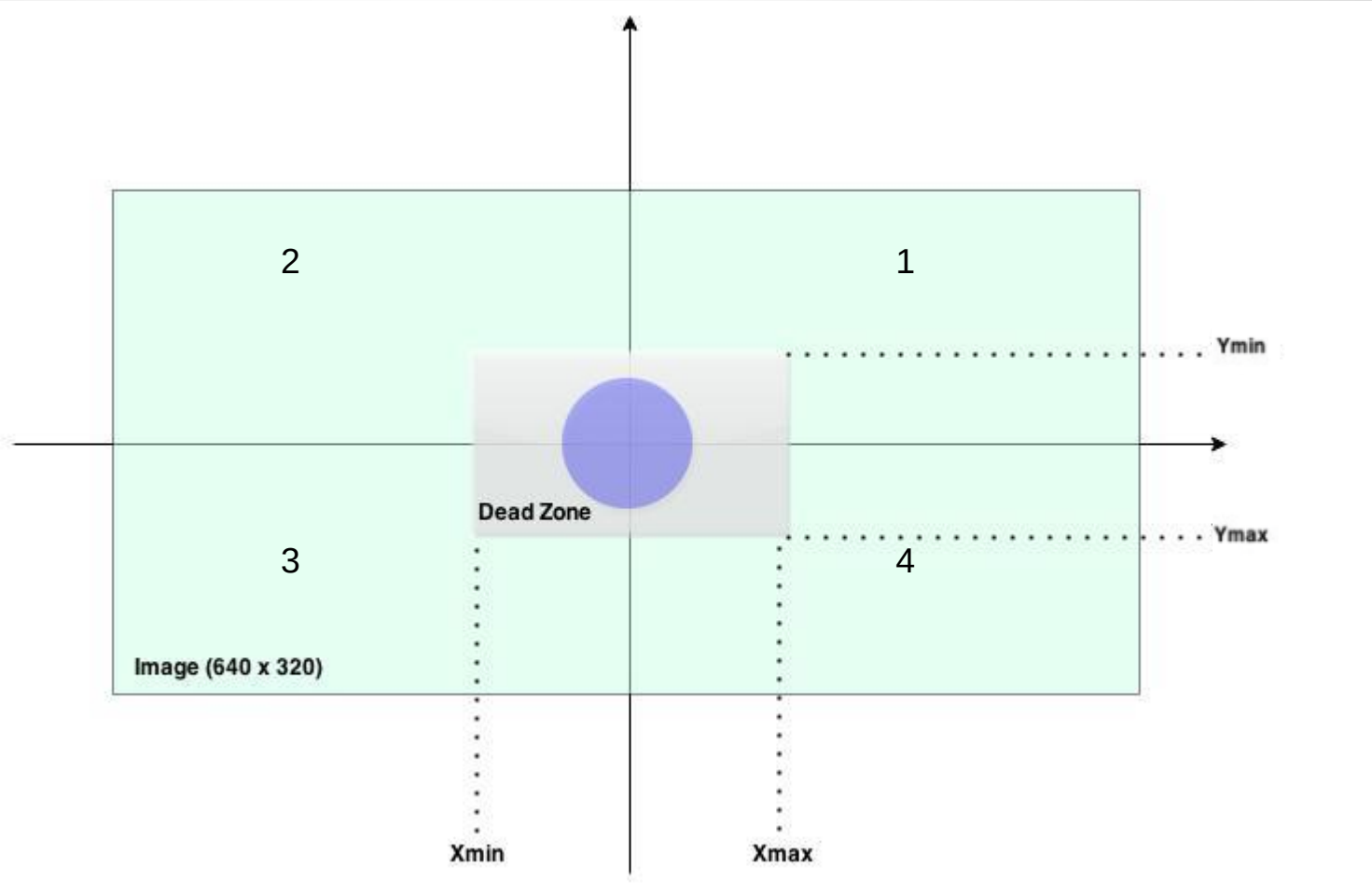
- A user can choose from three different colored balls to track
- When a ball is selected by the user, a message is sent to raspberry pi to indicate which HSV values should be utilized when performing the vision processing
- A non-blocking server is used to establish communication with the raspberry pi
- Each ball falls in a different range of hue values, allowing each ball to still be in frame

Vision Processing

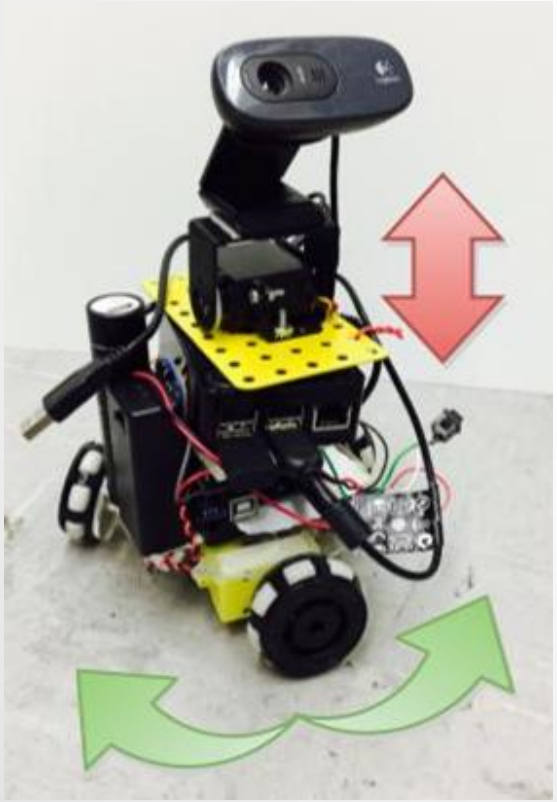
- An original frame(RGB) is converted to the HSV frame
  - An Input from iOS device is given to distinguish between different colors of the balls
  - For each input, the vision processing is performed to identify the contours of the ball (erode/dilate with HSV constraints)
  - The center of the ball is then calculated based on the identified contours
- The coordinates of the center is then compared with the pre-designated range
- Raspberry Pi sends a message to Arduino whether to move the motors to align the center of the ball with the center of the image

Vision Processing

Vision Processing



Arduino: Actuating motors



- Vertical movement
  - Initial Servo angle: 45 degree
  - Raising the camera: +1 degree every time when raise message received
  - Lowering the camera: -1 degree every time when lowering message received
- Horizontal movement
  - Clockwise rotation
  - Anti-clockwise rotation

Arduino: Actuating motors

Arduino Code

```
    if(t1 == -1 && t2 ==  1){
      pos +=  b;
      servo1.write(pos);
for (int j = 1; j<=A; j++){
       motor1.run(FORWARD);
       motor1.setSpeed(w1_speed);
       motor2.run(FORWARD);
       motor2.setSpeed(w1_speed);
       motor3.run(FORWARD);
       motor3.setSpeed(w1_speed);
   }
   motor1.setSpeed(0);
   motor2.setSpeed(0);
   motor3.setSpeed(0);
       delay(40);
    }
    else if(t1 == 1 && t2 ==  1){
      pos +=  b;
      servo1.write(pos);
 for (int j = 1; j<=A; j++){
       motor1.run(BACKWARD);
       motor1.setSpeed(w1_speed);
       motor2.run(BACKWARD);
       motor2.setSpeed(w1_speed);
       motor3.run(BACKWARD);
       motor3.setSpeed(w1_speed);
   }
   motor1.setSpeed(0);
   motor2.setSpeed(0);
   motor3.setSpeed(0);
      delay(40);
    }
```

....

Problems

- Slow reaction to the change
  - Due to the image processing being performed on the Raspberry Pi, there is a significant delay between new images and messages sent
  - Arduino processes the data much faster than Raspberry Pi's vision processing
  - Overshoot happens due to this delay
    - Which introduces oscillations
- Streaming / SSH to monitor the process in real time introduces significantly larger delays
- Standard servo motors conflict with the timer used in software serial
  - This causes for flickering in the servo pitching the camera

Conclusion

- Utilizing the Raspberry Pi 2’s computing capabilities, we were able to successfully track the motion of the selected ball
- Implementation of different advanced mechatronics topics allowed us to;
  - Create user defined variables for sensor input
  - Track the position of different colored objects utilizing a webcam
  - Use position data to operate a series of servo and DC motors in order to maintain object visibility
  - Communicate between different micro-controllers