EE109 Final Project

FPGA Car Control

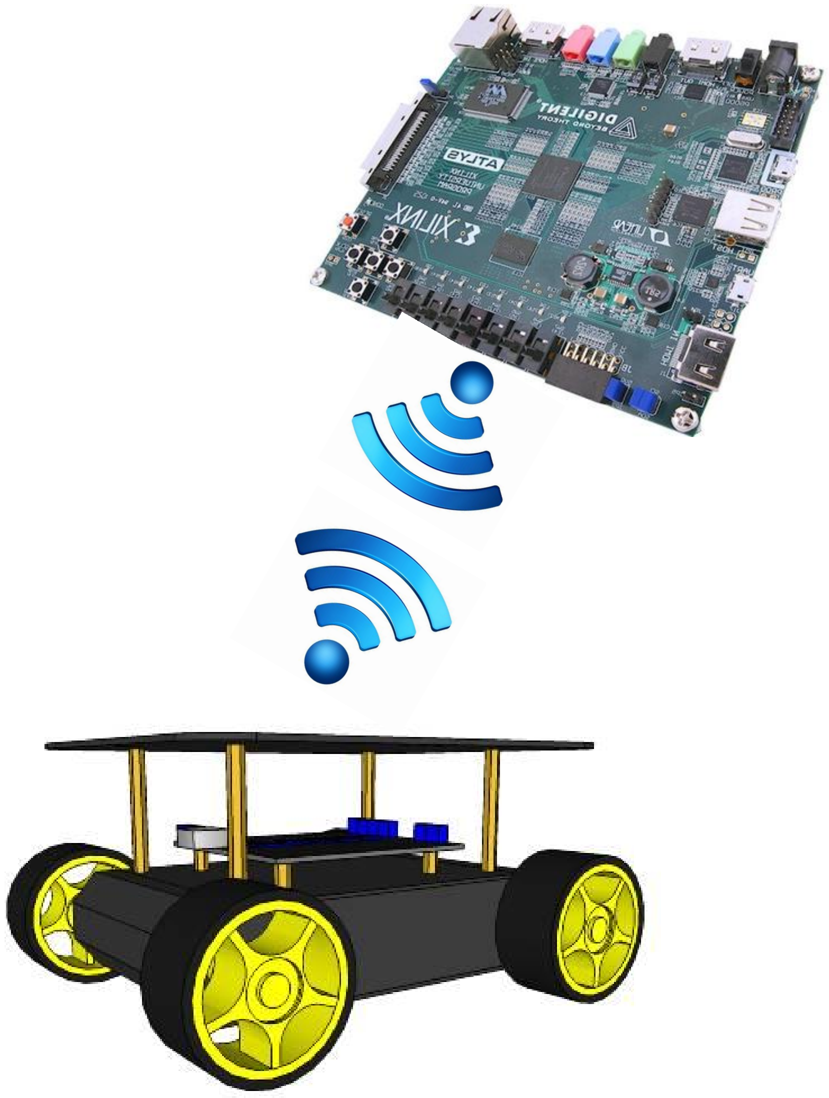
Tian Zhao
Neel Parikh

Professor Kunle Olukotun, Chris Copeland (TA)

Objectives

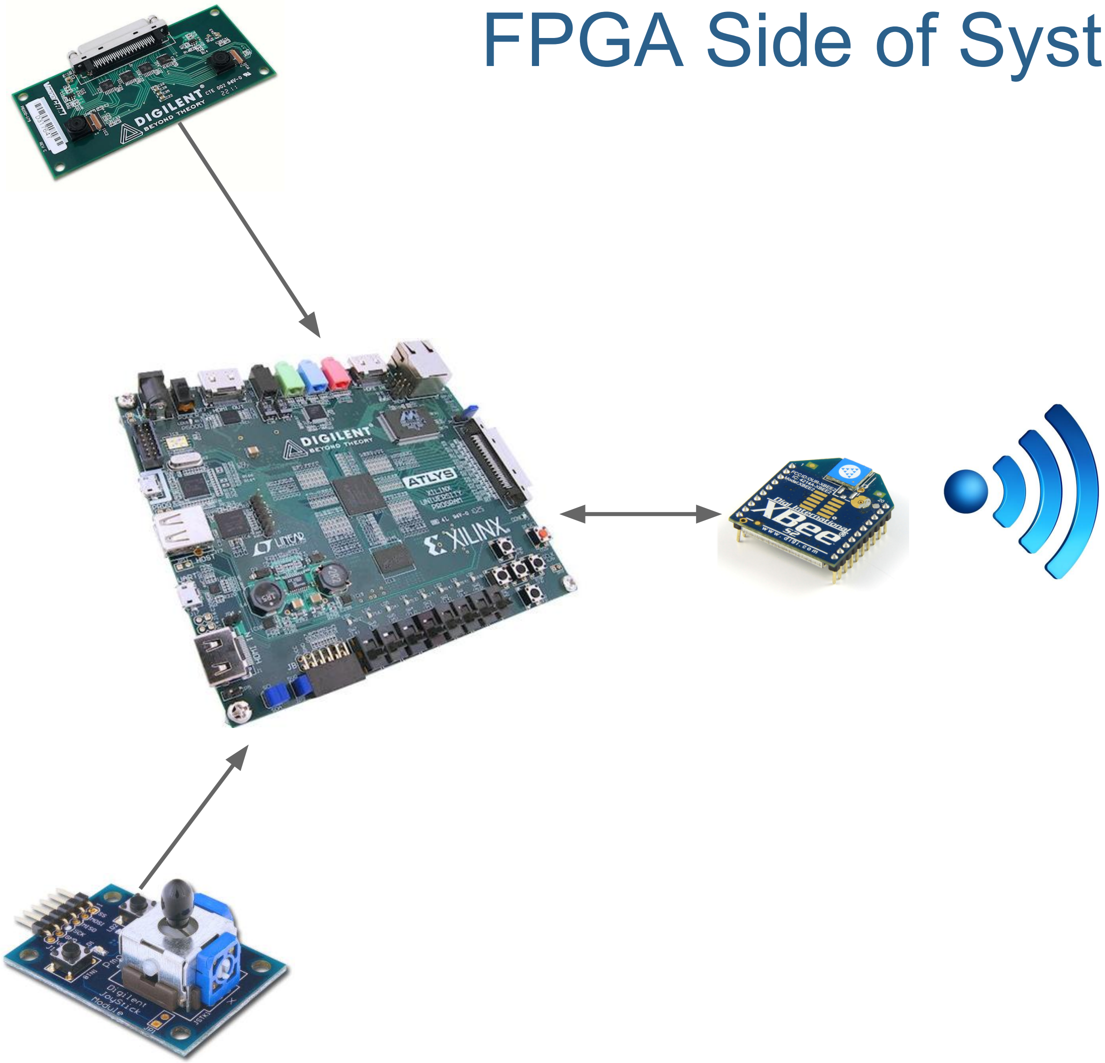
FPGA Controlled Vehicle

- Bidirectional wireless communication
- Manual user control
- Autonomous operation
- Visual feedback to detect position and orientation



Hardware System Overview

FPGA Side of System

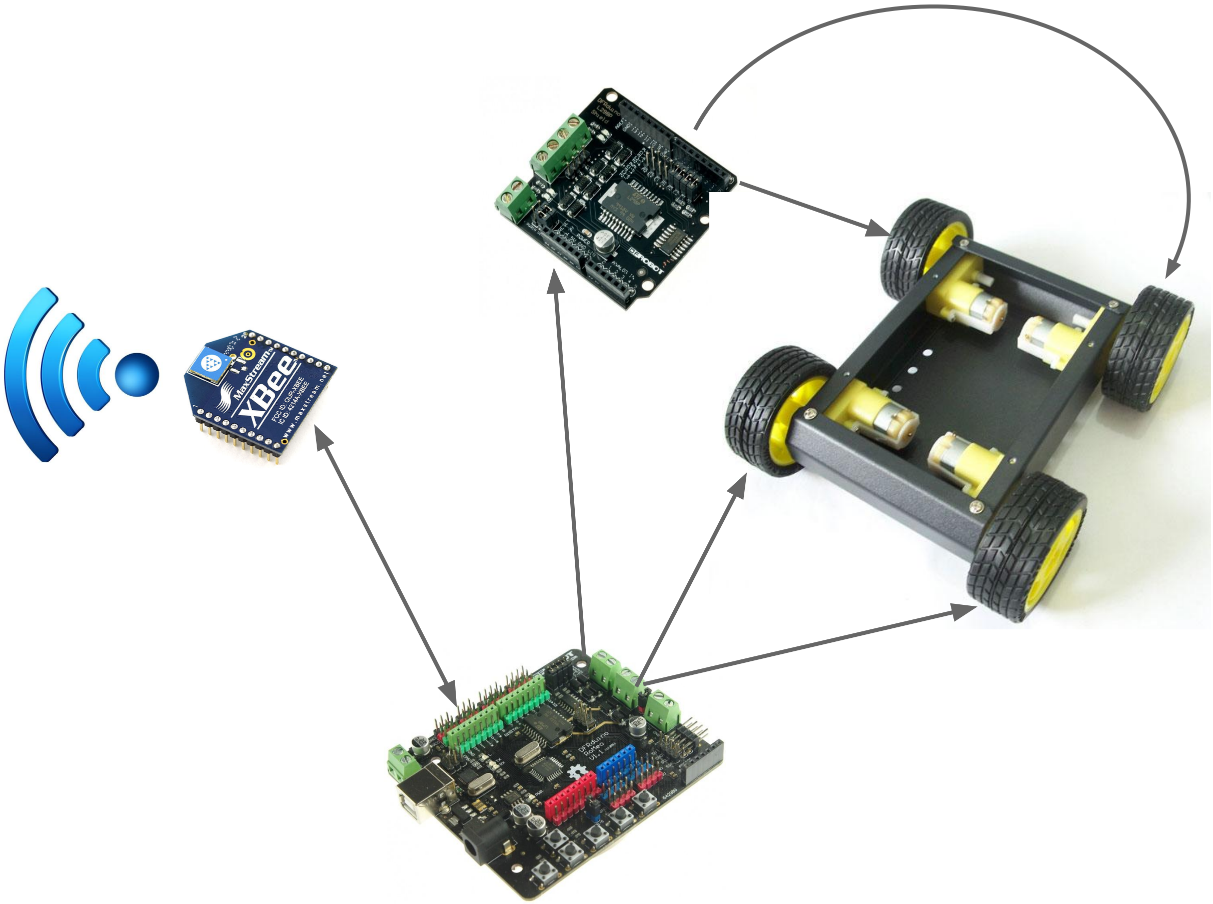


- Atlys FPGA board
- Vmod camera
- Pmod joystick
- XBee radio module

Hardware System Overview

Vehicle Side of System

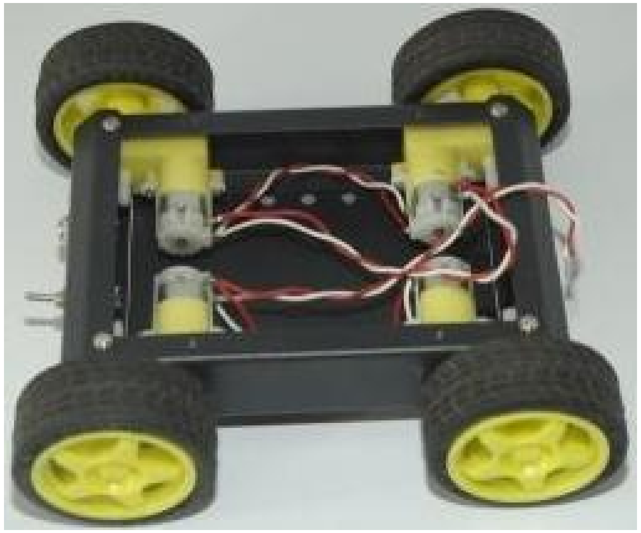
- 4WD vehicle platform
- XBee radio module
- Arduino board
- Motor control "shield"

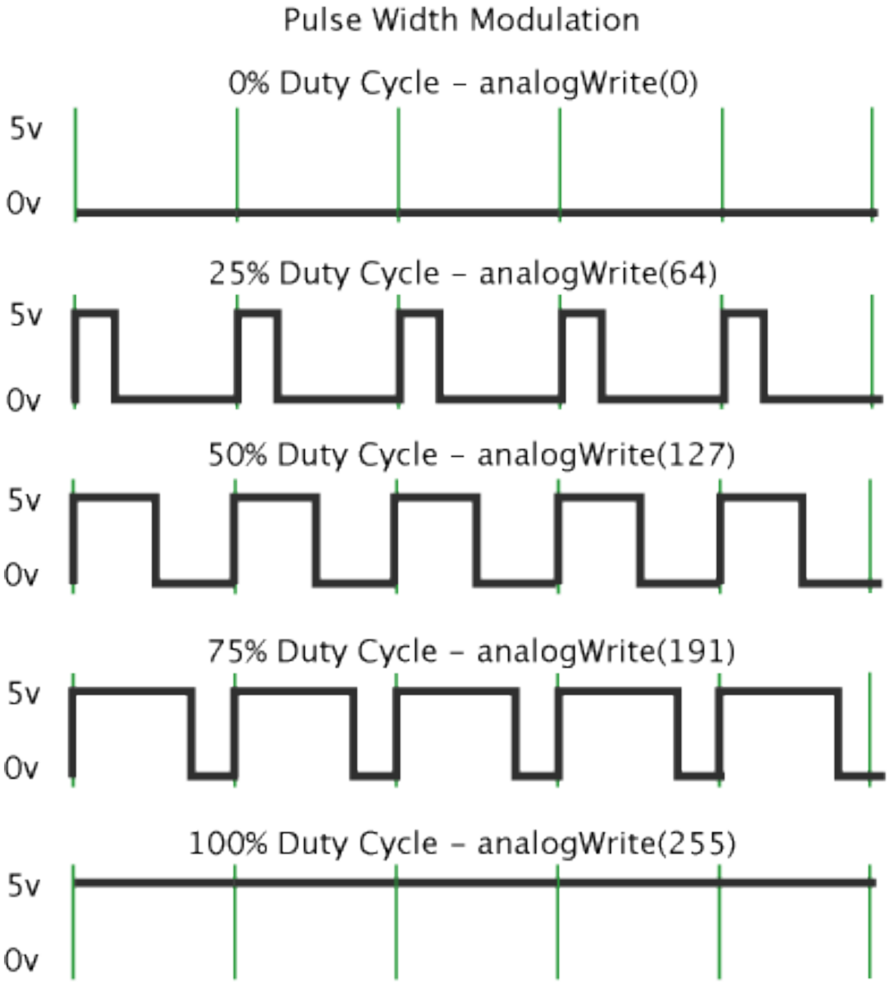


Vehicle Platform

Four DC Motors

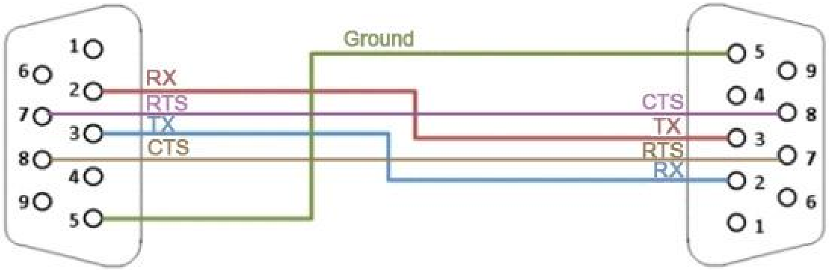
- Left and right pairs of wheels
- Direction of rotation controlled by power supply polarity
- Magnitude controlled by PWM





Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

Bidirectional Wireless Communication



Byte Oriented Serial Communication

- XBee connects to Atlys by USB - UART
- stdout and stdin piped to/from Atlys UART port
- bytes sent with *xil_printf()* and read with *getchar()*
- XBees operate in "transparent mode" using IEEE 802.15 (Zigbee)

Modes of Operation

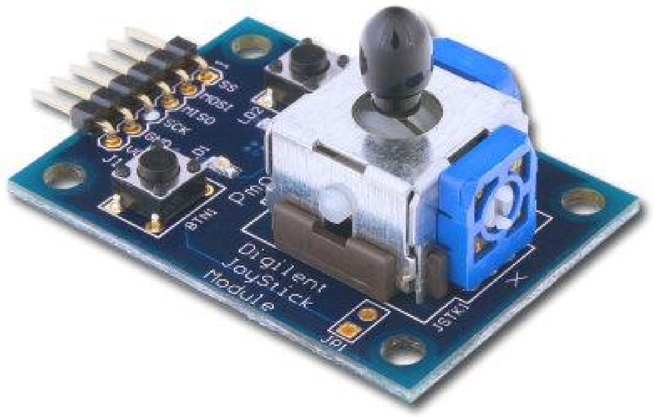
Three User Selectable Modes

- **Joystick Mode**: manual control by user
- **Command Mode**: execute programmed command sequence
- **Visual Mode**: commands regulated by camera input

Modes Selected by DIP Switches

- System behavior determined by software residing on both the FPGA and the Arduino

Manual User Control

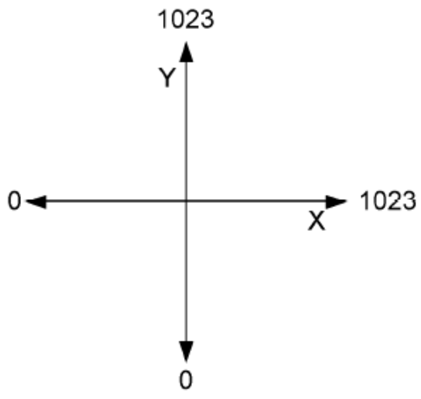


Joystick Mode

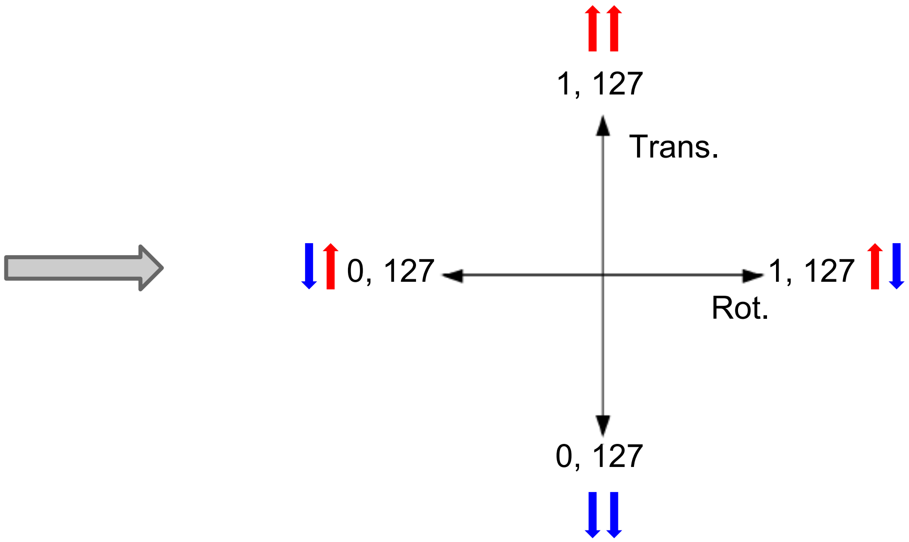
- Synchronous serial data link (SPI)
- Produces a steady stream of input

Data Processing

- Outputs two 10-bit values, X and Y position
- Memory mapped to 2 software accessible registers
- Algorithm converts X,Y to L,R direction and speed



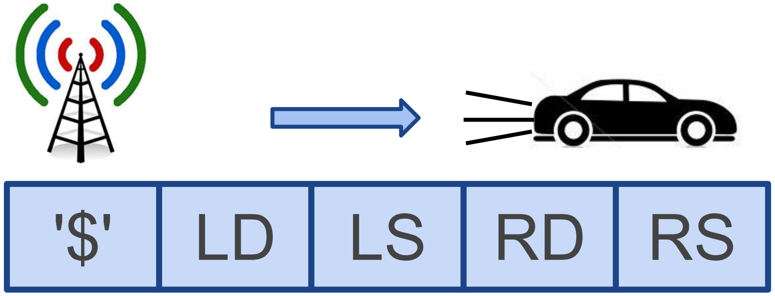
Joystick Axis Map



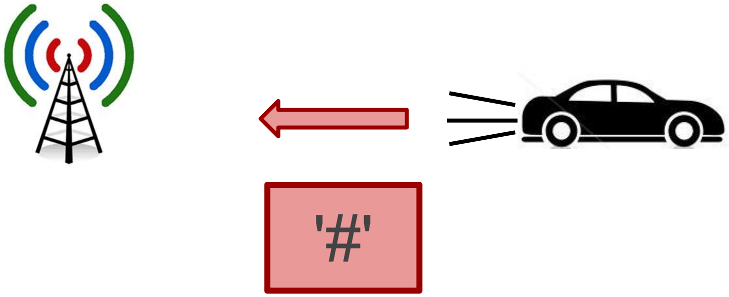
Joystick Mode

Transmitting Values to Car

- 5-byte control packet
- Packets sent in blocks of 5
- FPGA waits for request char from car



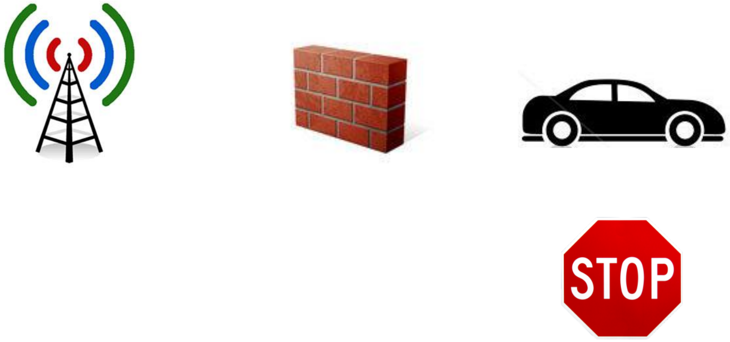
'$'	LD	LS	RD	RS



'#'

Fault Tolerance

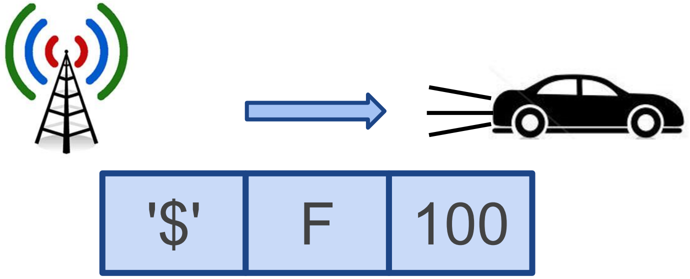
- Count a fault if input buffer has < 5 bytes
- Clear numFaults when packet received
- If numFaults > maxFaults then StopCar()



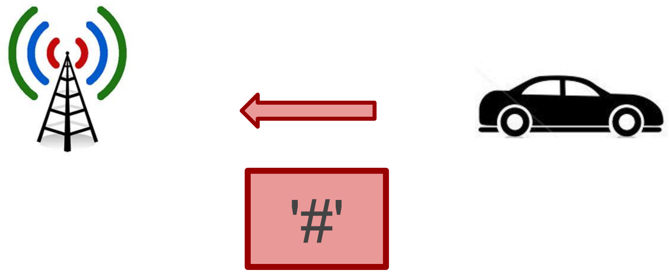
Semi-Autonomous Driving

Command Mode

- User programs sequence of commands
- Transmitted to car sequentially
- Pressing push-button repeats sequence



'$' | F | 100



'#'

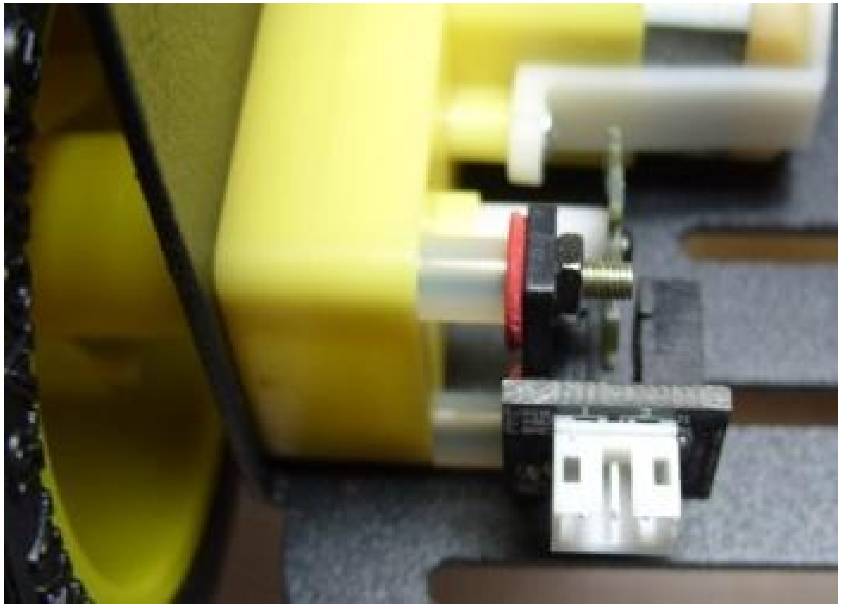
Transmitting Commands

- 3-byte control packet (command, amount)
  - 'f' - forward
  - 'b' - backward
  - 'l' - left
  - 'r' - right
  - 'w' - wait
  - 's' - standby
- Packets sent one at a time
- Wait for request char after every command
- Only repeat request if numFaults > maxFaults

Command Mode

Wheel Encoders

- Allows discrete changes in position
- Blocks/unblocks pair of IR sensors
- Every change causes interrupt
- Handler counts interrupts
- 20 interrupts per rotation



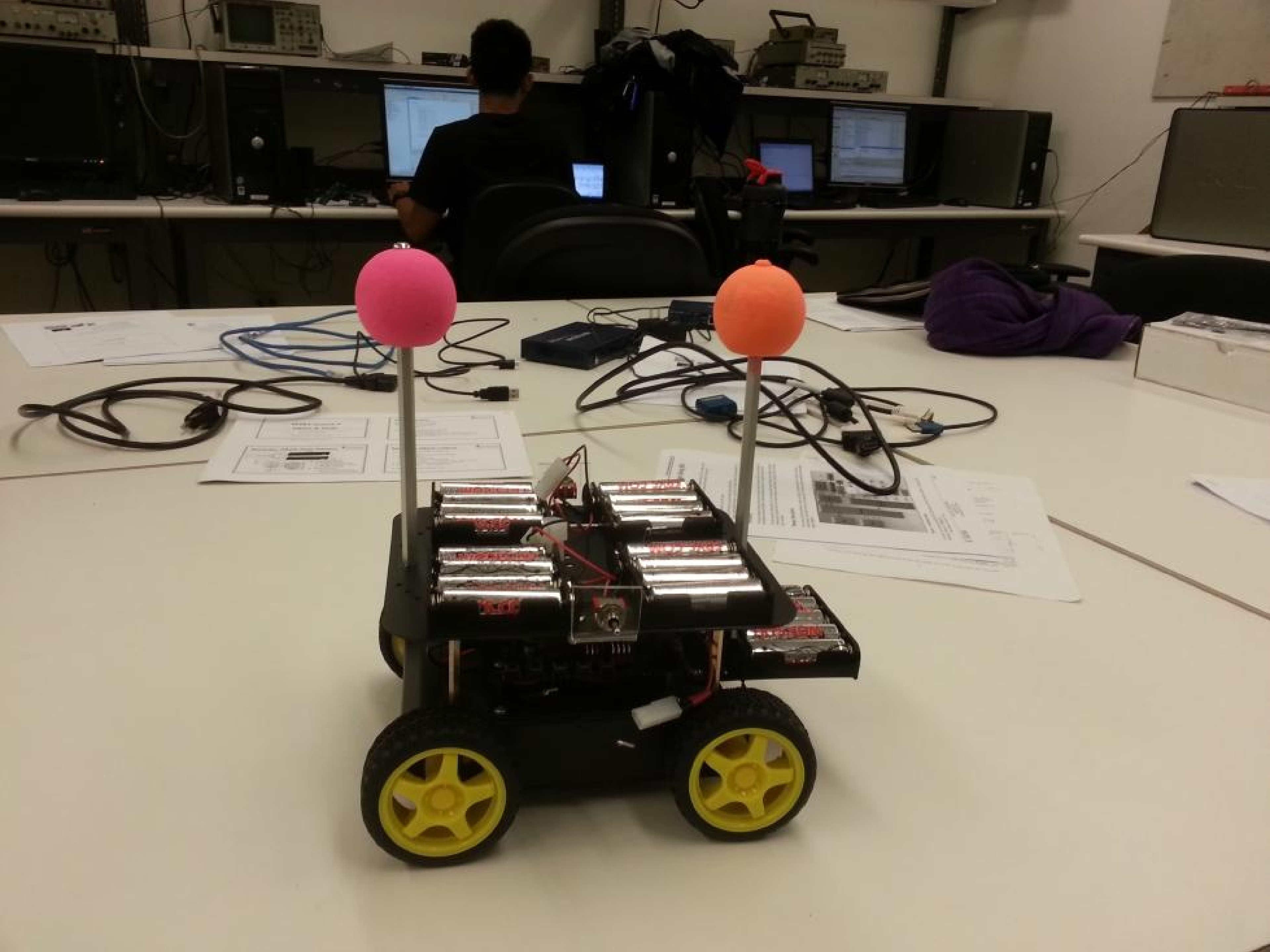
Visual Feedback

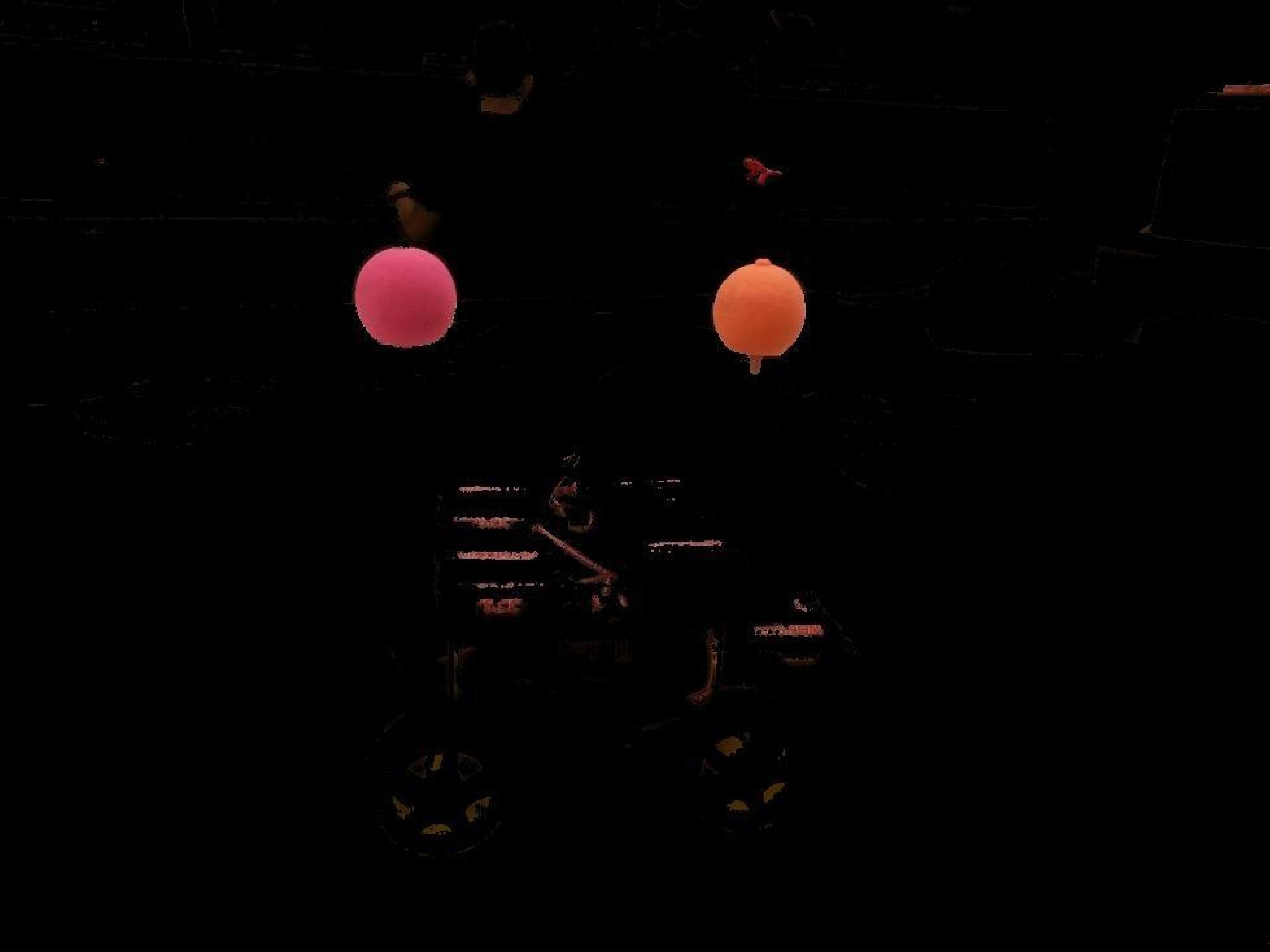
Image Filtering

- Each pixel from current display frame tested
- Computational efficiency important for speed
- Level1 filter removes about 90% of pixels

Approximately **ORANGE** If:

$$R > 1.5 * G \;\&\&\; G > B$$



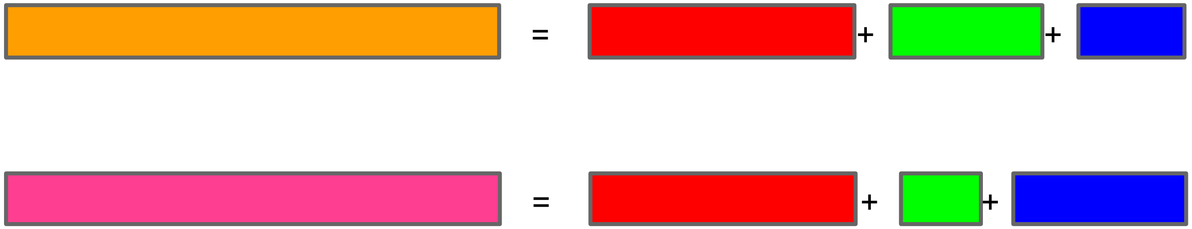


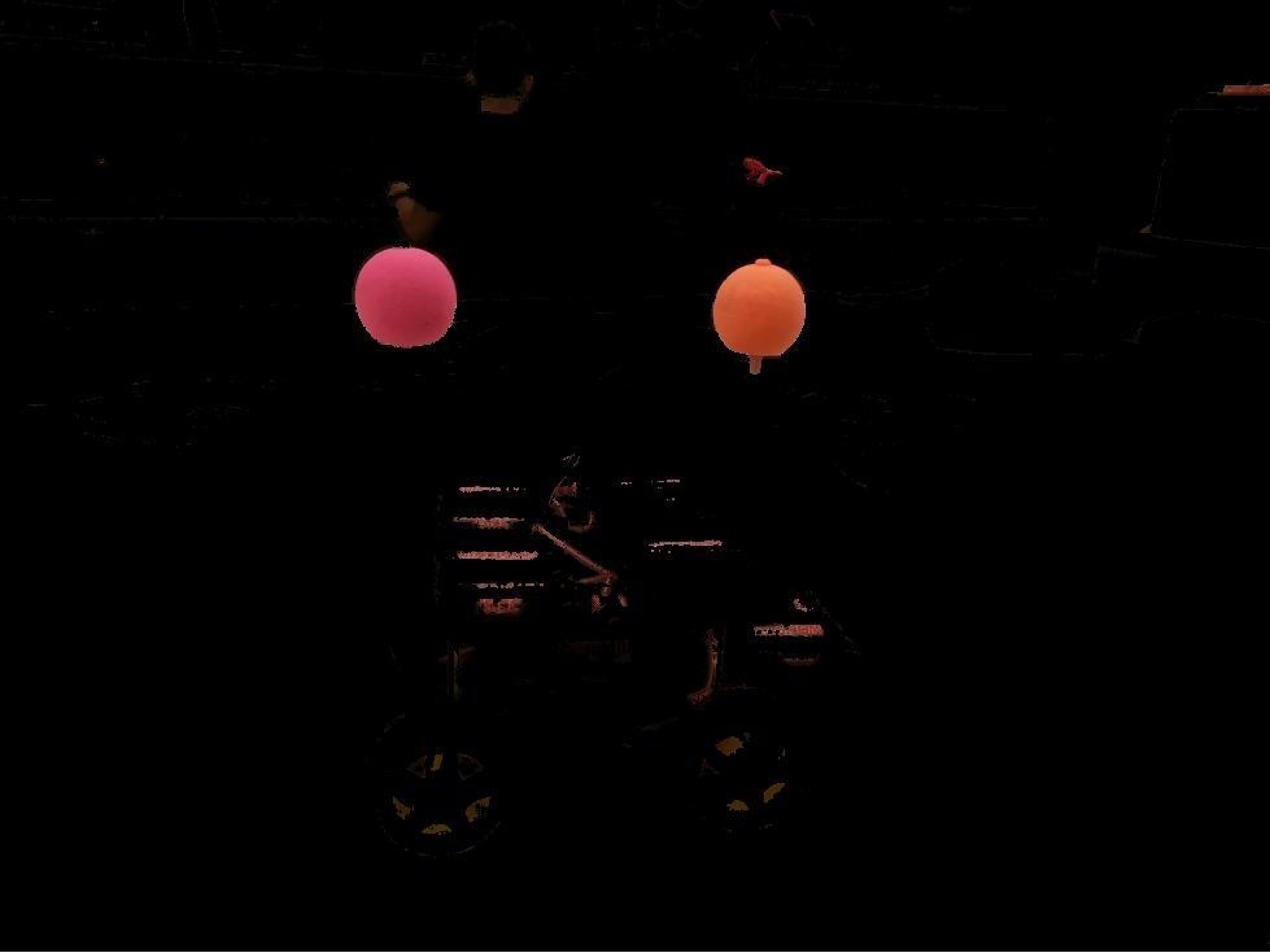
Visual Mode

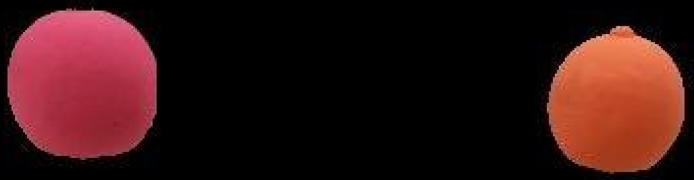
Level 2 Filter

- Calculates chromatic distance from target color
- Uses component differences to approx. HSV
  - RG = **R** - **G**
  - RB = **R** - **B**
  - BG = **G** - **B**
- Distance must be less than chromaticDistanceLimit

Target Colors



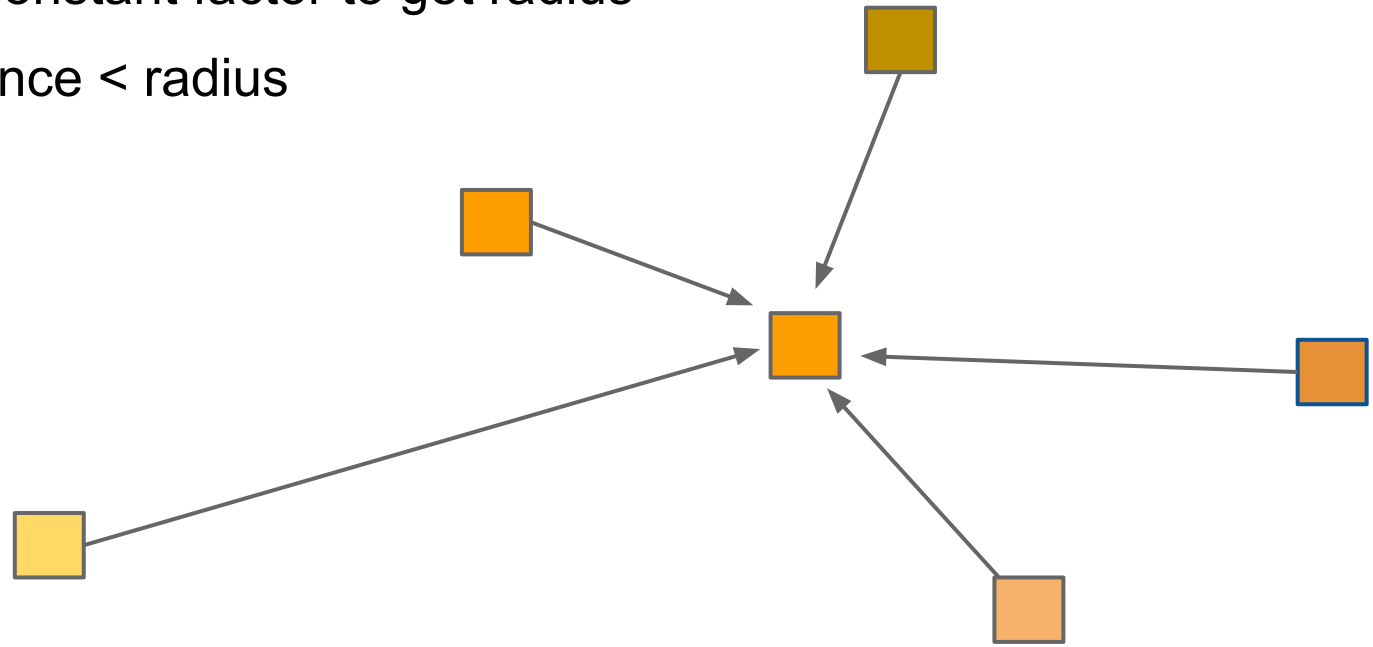




Visual Mode

Spatial Filter

- Used to detect position and size of markers
- Average distance to target position (x,y)
  - pixels that passed chromatic filter
- Multiply by constant factor to get radius
- Pass if distance < radius



Visual Mode

Adaptive Filtering

- newTargetColor = (2*BASE_COLOR + oldTargetColor + avgPixelColor) / 4
- newChromDistLimit = maxChromDist of all pixels that passed
- newTargetPos = average_X, average_Y
- newRadius = (oldRadius + avgRadius) / 2