Project Architecture

Looney Code

EC327 Final Project

Overview

Introduction

Solution

- Design
- Technical Components/Integration
- Challenges

Conclusion

Demo

Future Work

Looney Coders

Christian So

Project Lead, Software/Hardware Integration Lead

Beren Donmez

GUI Lead

Jiahe Niu

Control Systems Leads

Problem Statement

Everyday objects that **require physical contact to handle**, such as shopping carts, bathroom door handles, etc, can accumulate tremendous amounts of bacteria that can **increase the rate of transmission of diseases.**

FACT:
Grocery Store Shopping carts have **270x MORE** bacteria than your toilet handle. (Source: INSIDER)

Current Solutions

Solution 1: Wiping the handlebars of the shopping cart

Problem: Bad for the environment, most shopping markets do not have wipes available, only hand sanitizers.

Solutions 2: Covers for the shopping cart

Problem: Expensive, time consuming, inconvenient

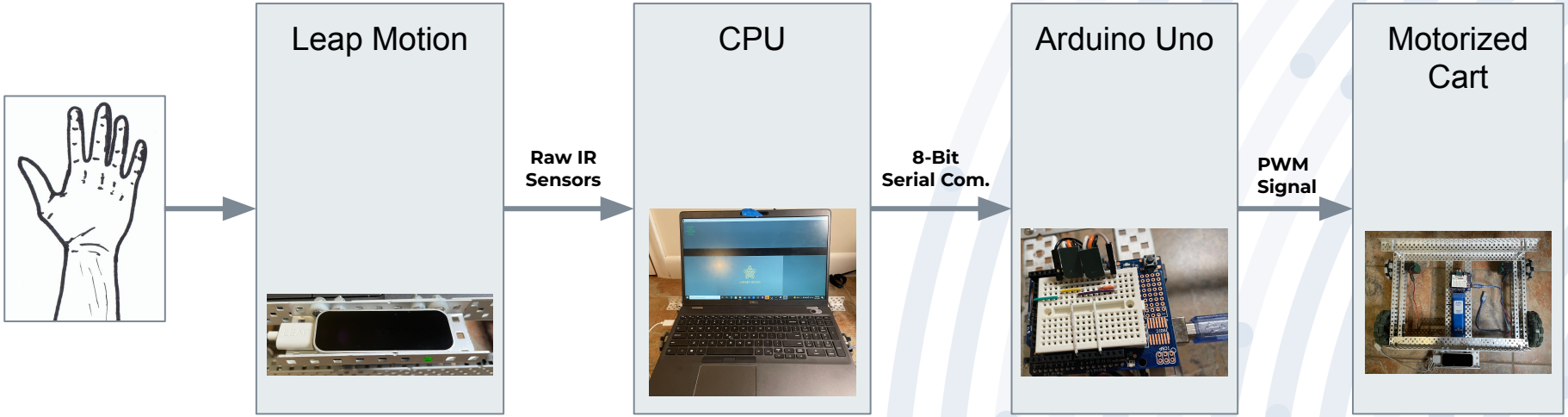
Marketable Solution

Solution needs to

- Dramatically decrease spread of germs without additional supermarket employee manpower
- Reusable
- Low-cost (time and money)
- Convenient

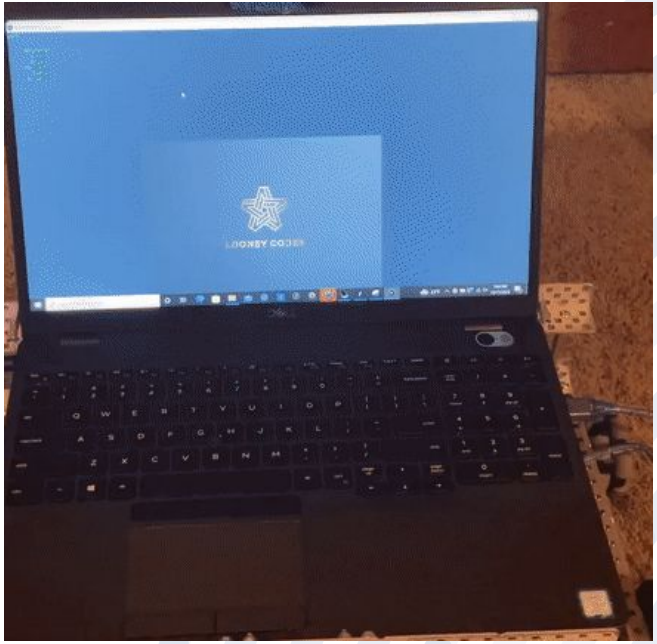
Solution

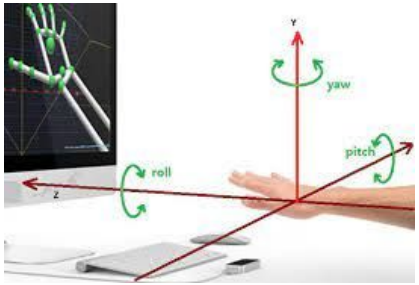
Contactless shopping cart that controls **motorized wheels with hand gestures,** mimicking grabbing and moving a shopping cart, without the handle. This solution further extends to any object that requires contact to move.

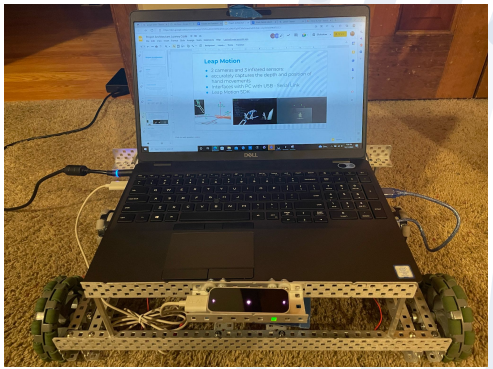


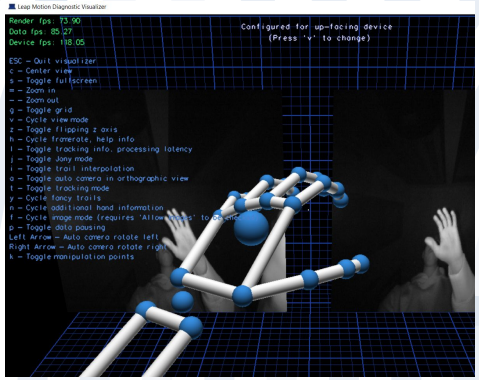
Leap Motion

- 2 cameras and 3 infrared sensors:
- accurately captures the depth and position of hand movements
- Interfaces with PC with USB - Serial Link
- Leap Motion SDK









CPU - Laptop

- Leap Motion SDK
- Processing written in Java for hand visuals and hand data parsing
- Serial Communication - 8 bit
- Arduino IDE in C++
  - Used for data analysis and output hardware communication
  - PID Classes
  - Motor classes

Graphical Interface

Using **Adobe Express**, we designed a special screen that pops up each time the handle bar is grabbed. The screen consists of the message "**Grabbed**", an image of a hand which grabs the handle, and our logo.

Declared the logo and hand grab image as

-PImage image;

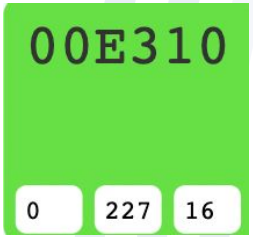
-PImage handGrab_pic;

Function to load the images:

-loadImage("");

The color declarations:

-fill(#00E310); //three dots as hand is detected



Screen on SLIDE 10

Graphical User Interference

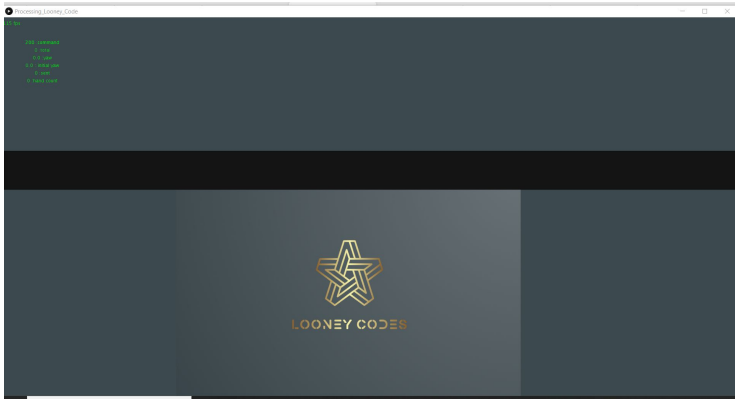
Initially, int flag is set to 1
-Hand is not grabbed in the Beginning
if(flag==0 && hand Count==1)
-flag 0 indicates the hand is grabbed
-handcount is detected, it executes the if statement where we have the "Grabbed" statement
Inside this if statement we have another if statement

if(send<200): the value is less than 200, not a command, so the picture will move
else:
- it will detect a command and the hand will be kept in the middle

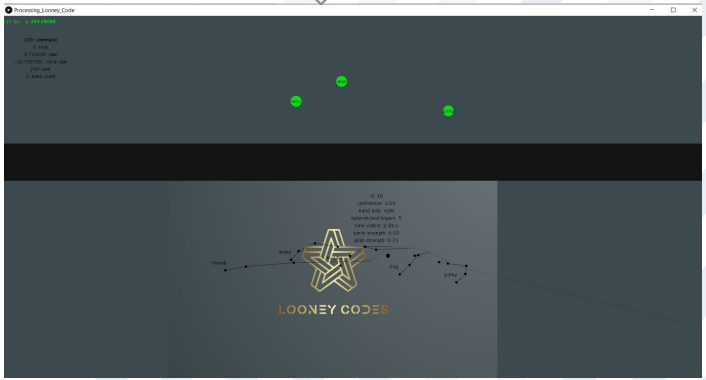
image(img, a, b)

Parameters

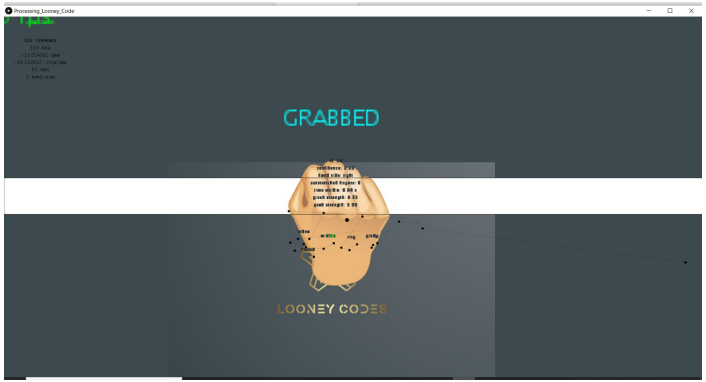
- **img**(PImage):he image to display
- **a**(float):x-coordinate of the image by default
- **b**(float):y-coordinate of the image by default



Hand Detected



Grab Gesture Detected



Processing Code

setup() + global parameters
- Sets up initial parameters
  - **200: Stop (initial state)**
  - 201: Forward
  - 202: Backward
  - 203: Left turn
  - 204: Right turn

Initial angle along y axis (yaw) is stored when hand is grabbed and then compared to movement of hand which changes command code

draw()
- Gray handlebar is drawn
- Hand (ang finger joints) are drawn when appearing in the sensor

for(Hand hand : leap.getHands ()
After the hand is grabbed:
- Handlebar turns white
- Initial position of hand is stored
- Additional movement is compared to initial

Command is sent to Arduino to continue processing

Arduino_Looney_Code.ino

Arduino Code

Main Arduino file

- creates objects Motor and PID
- Checks if serial communication is available
- Reads from serial using Serial library
- Serial values are truncated to 100 if over 100, numbers above 100 are reserved for commands
- Calls object functions and sends signal to Arduino Uno

Arduino_Looney_Code.ino

Arduino Class Objects

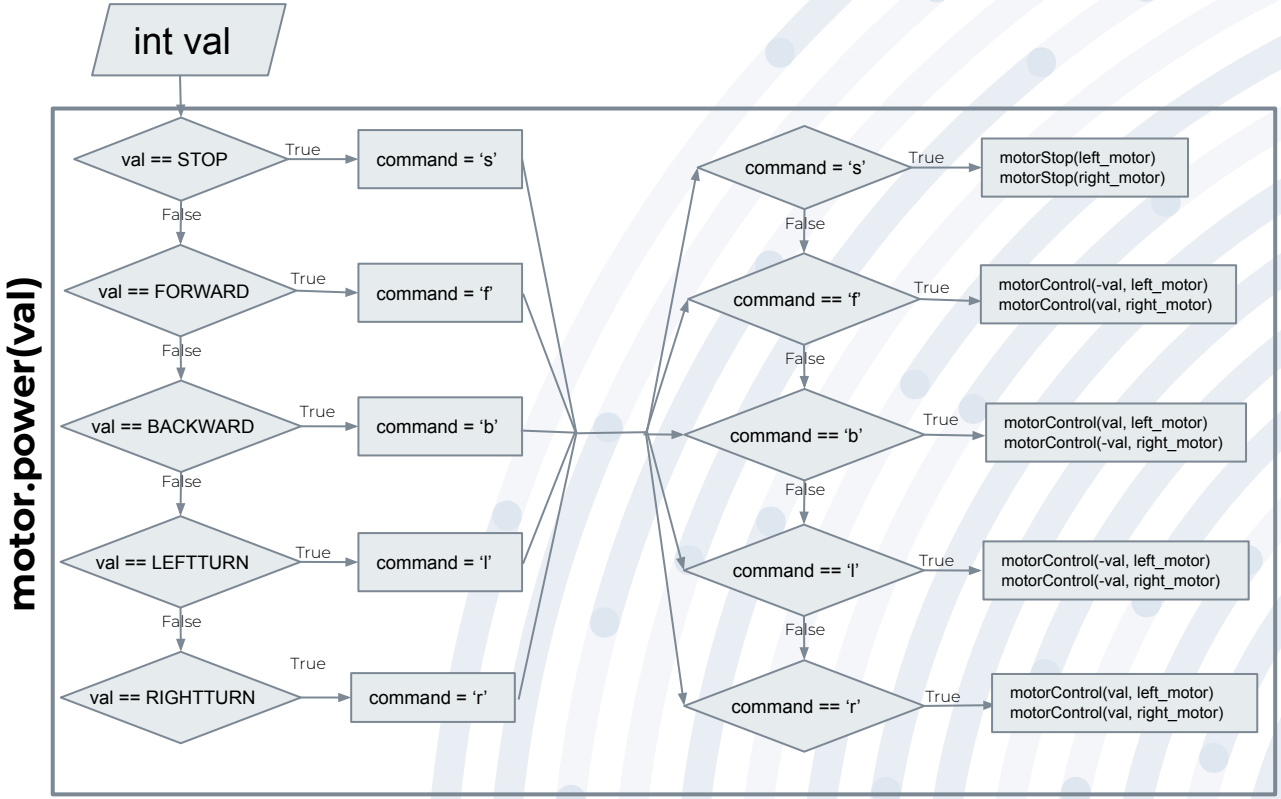
Motor Class
- char command;
- Servo left_motor
- Servo right_motor
+ void set_pins(int right_pin, int left_pin)
+ void motorControl(int value, Servo motor)
+ motorStop(Servo motor)
+ void power(int val);

PID Class
- double kP
- double kI
- double kD
- int computed_speed
- int prevError
- int total error
- int derivative
- int Pvalue
- int Ivalue
- int Dvalue
- unsigned long currentTime
- unsigned long previousTime
+ int target
+ int value
+ PID(double P, double I, double D)
+ PID(double P, double D)
+ int ComputeMotorSpeed(int val)
+ void updateTime();
+ void setP(double val)
+ void setI(double val)
+ void setD(double val)
+ double elapsedTime

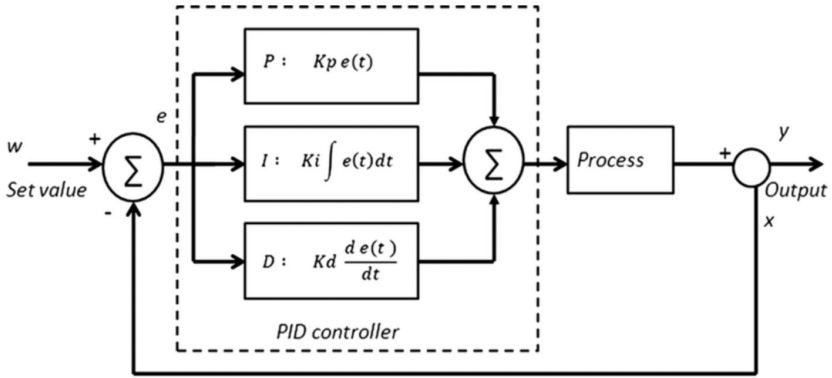
- Classes were created for **Modularity**
- Allowed for **different hardware configurations** with minimal software changes
- **Different tuned values of PID** for turning and straight movement

Motor Class

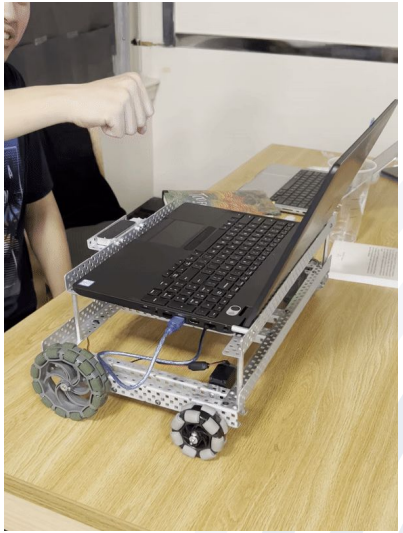
- **Servo library** to produce PWM signal
- **set_pins function** defines output pins on Arduino Uno board
- **Mapping function** to hack 393 VEX motor
  - -100 to 100 mapped to 1000 to 2000 for Servo library
- **power() function** handles the different modes: forward, backward, left and right turning
  - Determined by comparing **command codes to input values (on right)**



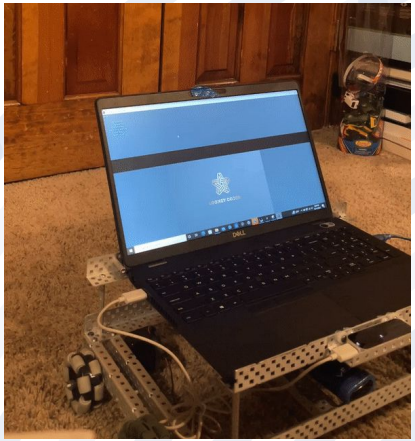
PID Class



Before PID



After PID

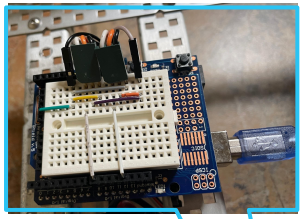


PID was implemented at first, however, PD control was used because we wanted our system to mainly focus on responding to sudden changes (hand movements) rather than have good error tracking.

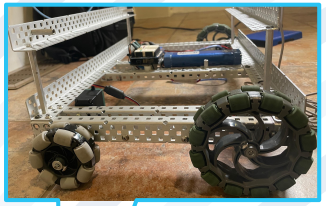
Motorized Cart

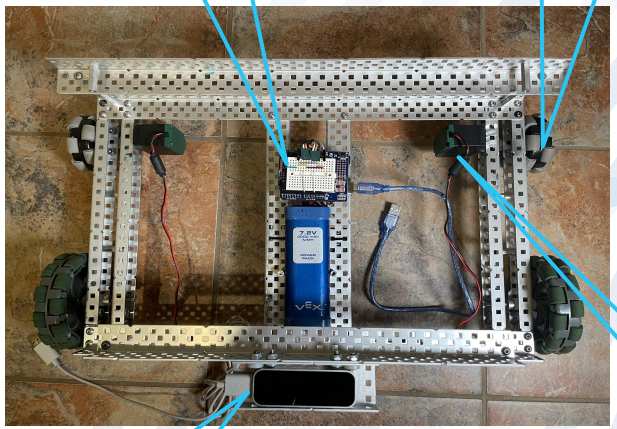
- Motor control - takes in 7.2V and **pwm signal** from Arduino Uno
- **393 Motors(100rpm)**(x2) attached to 2.75in wheels for torque and reduce jerk
- **Aluminum base** to minimize dead load and comply with motor power
- **Leap Motion attached upwards** to moving cart - minimize background noise and unwanted hands

Arduino Uno w/Circuit

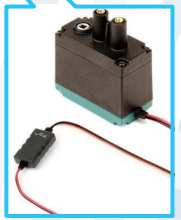


2.75in(front) and 4in(back) Omni Wheels





Leap Motion Sensor



VEX 393 Motor w/motor controller

Integration

- **Initialize hand position** when it detects grab(mimicking grabbing a handle)
- **Map X,Y, Yaw difference** values from -100 to 100 to comply with serial communication
- Arduino read from serial and computes motor power value **using a tuned PID class object.**
- Arduino maps values to custom values for 393 motors using a Servo library and sends PWM signal
- Cart moves until the **hand is back in center**

Challenges

- Learning Java and Leap Motion Libraries and other libraries
- Hacking a VEX Motor to work w/Arduino
- Value resolution with 8-bit serial communication
  - Can’t send negative numbers
- Jerk and oscillation
  - Real world is not ideal

Challenge's Solutions

- Learning Java and Leap Motion Libraries and other libraries
  - **Solution:** Studying Documentations
- Hacking a VEX Motor to work w/Arduino
  - **Solution:** PWM from Servo Library
- Value resolution with 8-bit serial communication
  - Can't send negative numbers
  - **Solution:** with specific state codes
- Jerk and oscillation
  - Real world is not ideal
  - **Solution:** PID/PD control

Demo Video

Future Work

- Two Hand control
- Cheaper sensor
  - Raspberry Pi Camera
- Modular base to attach wheels to other objects
- Better sensor to hardware communication
- Smaller CPU
- Hardware log(motor speeds, hand detection, etc)
- Sensor Filtering
  - Kalman Filtering