ROBOTIC COMPUTER VISION

Sean Morton, Northwestern University
Anirudh Adiraju, Vernon Hills HS

Theme: "Geometric Electricity" by SlidesGo

Northwestern | McCORMICK SCHOOL OF ENGINEERING

01

OBJECT DETECTION SOFTWARE

Using MobileNet SSD Neural Net

Convolutional Neural Network

MobileNet SSD: An object detection model optimized for near-instantaneous speeds.

Performs “Single-Shot Detection” - more efficient at analyzing an image than R-CNN or YOLO algorithms

First developed in 2015: “SSD: Single Shot MultiBox Detector”, Wei Liu, D. Anguelov et al, Cornell university

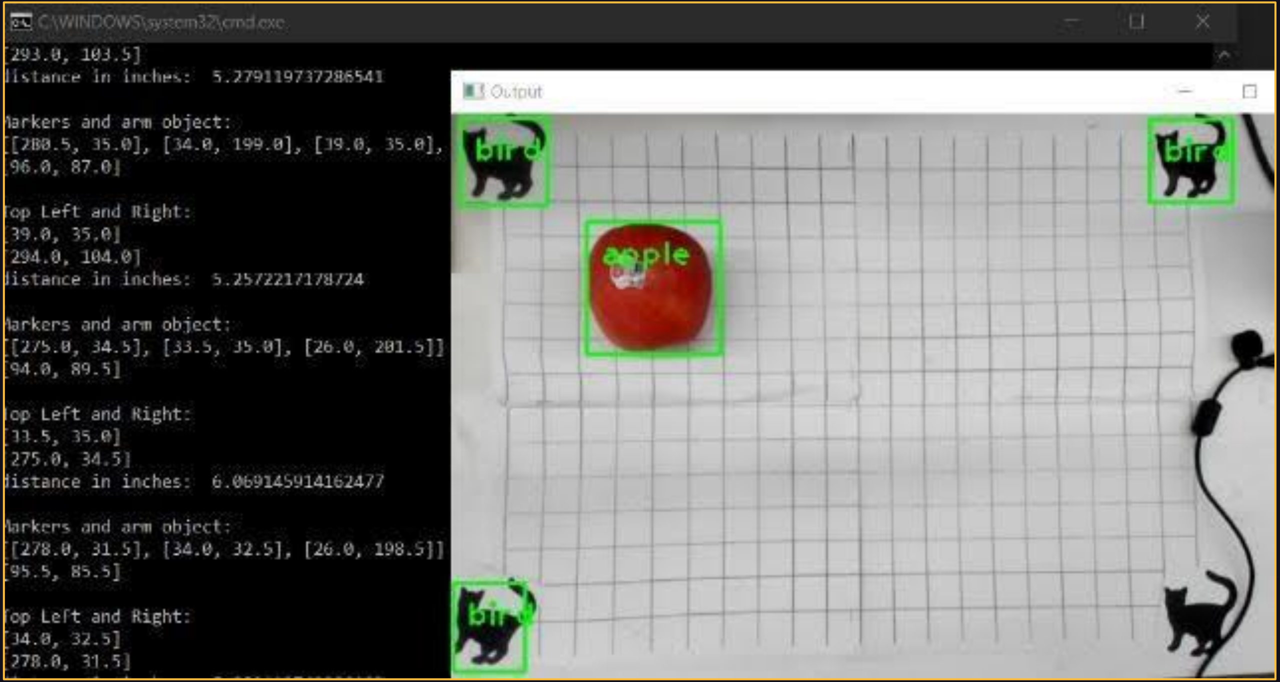
SSD is faster, but less accurate than R-CNN, and faster and more accurate than YOLO algorithms.

Object Detection Model

- Neural net trained on 90 classes
  - Many of the classes relate to self driving vehicles
- Classifies objects and determines bounding boxes (coordinates within image)
- Very fast with over 40% confidence
  - Speed and accuracy makeup for other flaws
- Future application: classification prediction affects robot arm functions
  - Ex: softer objects in the dataset, like bananas, require less grip strength to pick up

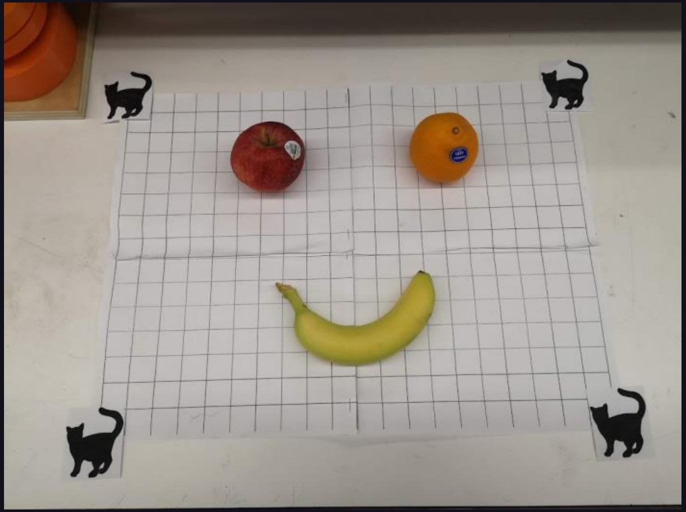
Object Detection Results

- Accurate prediction of distance in real life
- Works with different positions on the grid
- Markers classified as bird or cat

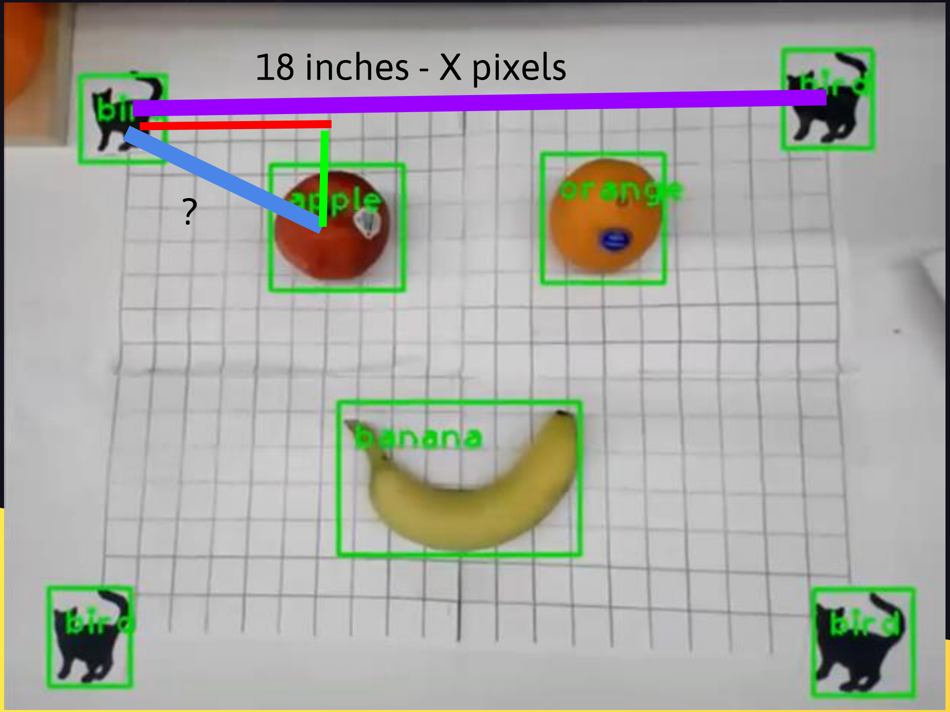


Detecting Real-World Position

1. Distance between fiducial markers is measured and kept consistent
2. Center coordinates of markers are found
3. Center coordinates of object are found
4. Distance in pixels is found using distance formula
5. Proportions are used to convert distance in pixels to distance in real life
6. Commands are sent to robot to move arm and pick up object



Diagram/Code




```
print("\nTop Left and Right:")
print(top_left)
print(top_right)


try:
    x_squared_ref = (float(top_right[0]) - float(top_left[0]))**2
    y_squared_ref = (float(top_right[1]) - float(top_left[1]))**2
    conversion = 18/(x_squared_ref + y_squared_ref)**0.5

except IndexError:
    print("Index error: conversion")

try:
    x_dist = (float(arm_object[0]) - float(top_left[0]))
    y_dist = (float(arm_object[1]) - float(top_left[1]))



    #multiply the distance by the conversion factor, pixels to inches
    x_dist *= conversion
    y_dist *= conversion
    print("x distance: ",x_dist)
    print("y distance: ",y_dist)

except NameError:
    print("Name error: conversion wasn't properly calculated")

except IndexError:
    print("Index error: calculating dist of object")
```

Interface with Arduino

A script for displaying text onto an LCD screen. This program was a test for using Python to send info to the Arduino Serial Monitor.

For this project, Python:

1. Takes an object classification
2. Finds the position of that object in real space
3. Uses Inverse Kinematics to calculate joint angles
4. Sends joint angles to the Arduino Serial

```python
import serial
import time
arduino = serial.Serial(port='COM4', baudrate=115200, timeout=.1)
def write_read(x):
    arduino.write(x.encode())
    time.sleep(0.05)
    data = arduino.readline()
    return data

concat = ""
obj = input("Enter an object: ") # Taking input from user
concat += obj

x_coord = input("Enter the x coordinate: ") # Taking input from user
concat += ","
concat += x_coord

y_coord = input("Enter the y coordinate: ") # Taking input from user
concat += ","
concat += y_coord
concat += ","
value = write_read(concat)
print(value) # printing the value
```

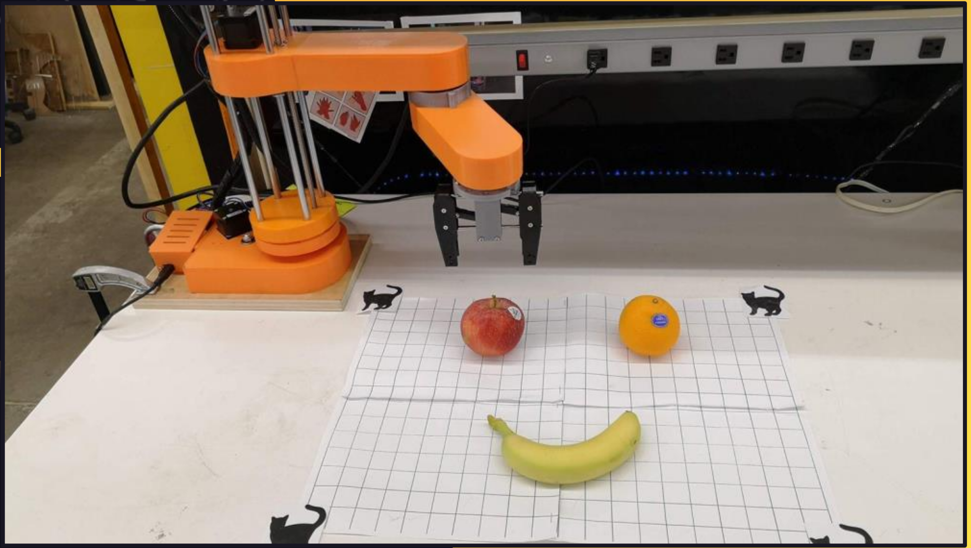
Interface with Arduino (cont.)

```
void loop() {
  if(Serial.available() > 0){
  lcd.clear();
  x = Serial.readString();
  Serial.print(x);
  int c = x.length();
  for (int i = 0; i < c; i++)
    {
      if(x[i] != ','){
        input += x[i];
      }
      else{
        if(count == 0){
          obj = input;
          input = "";
          count++;
        }
        else if(count == 1){
          x_coord = input.toFloat();
          input = "";
          count++;
        }
        else{
          y_coord = input.toFloat();
          input = "";
        }
      }
      }

    }
```

Video Demonstration

1.

2.



02

ROBOT HARDWARE

Mechanics of the SCARA Arm

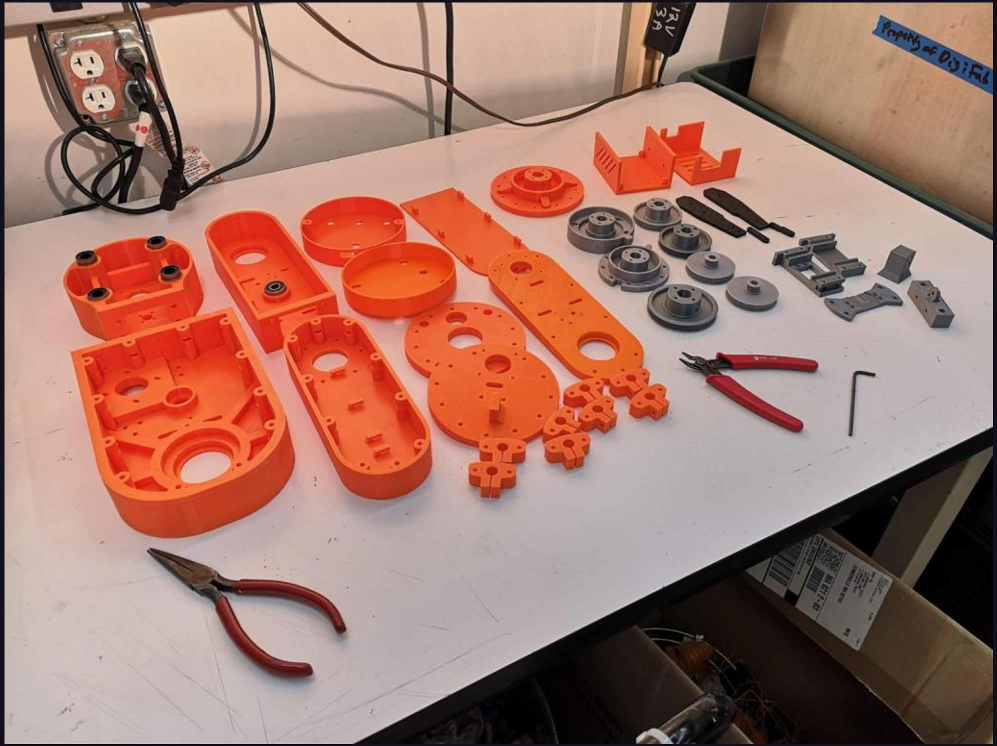
3D Printing Methods

Parameters:

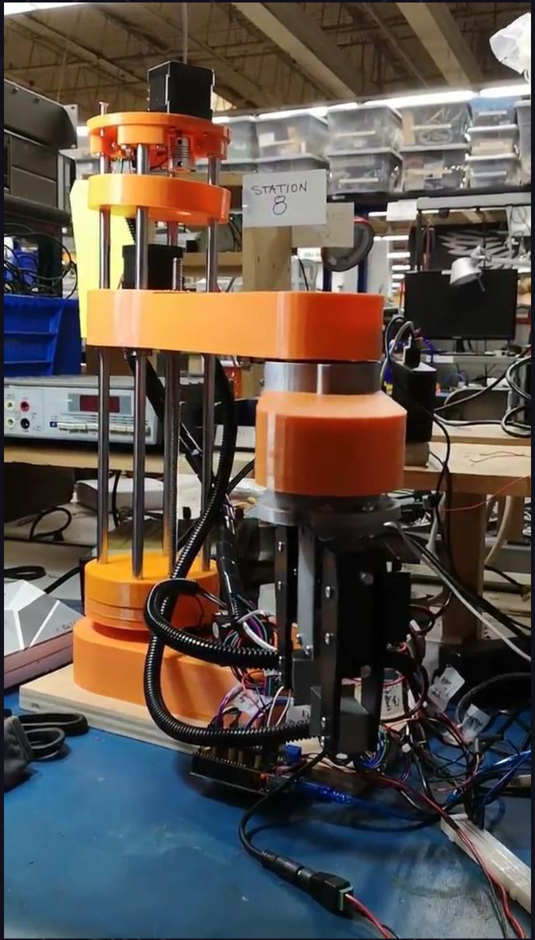
1. Flashforge Creator Pro
2. Hatchbox PLA/PETG 1.75mm filament
3. -0.1mm horizontal expansion compensation

Stats:

1. 120+ hours print time
2. 3 weeks worth of print
3. Several failed prints and broken printers encountered

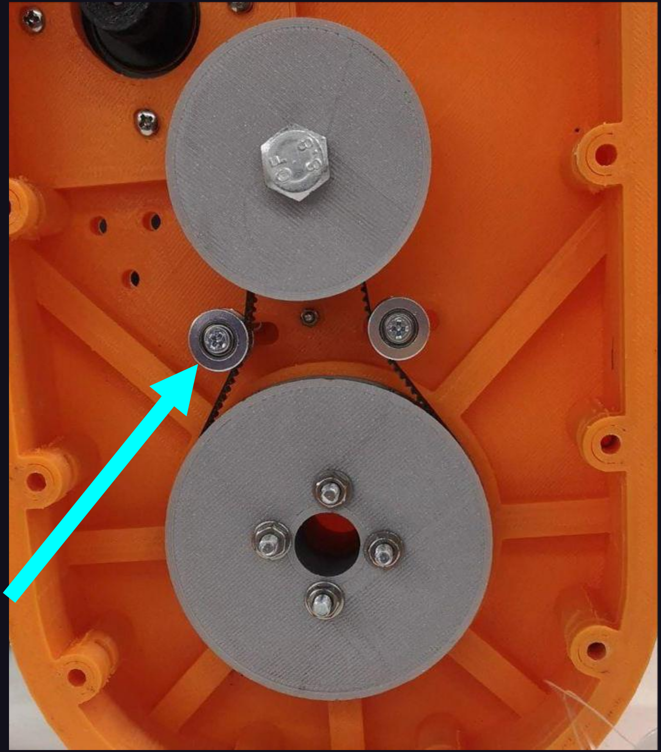


Robot Demonstration



Mechanical Components

Belt + Pulley System, w/ gear ratio



Belt Tensioners (highlighted)

Quick note: Artisan’s Asylum

Credits to Artisan’s Asylum, a workshop in Somerville, MA where I constructed this robot. Stations I visited:

Machine Shop:

- Mill + drill press used to drill precision holes
- Hand saw + lathe used to produce custom lengths of polished rod

Fiber Arts:

- Belts of approx. 200mm, 300mm, 400mm sewn by hand for pulleys

Wood Shop:

- Bandsaw used to cut wood plank for robot mount
- Drill press + countersink bit used for screwing robot to the platform

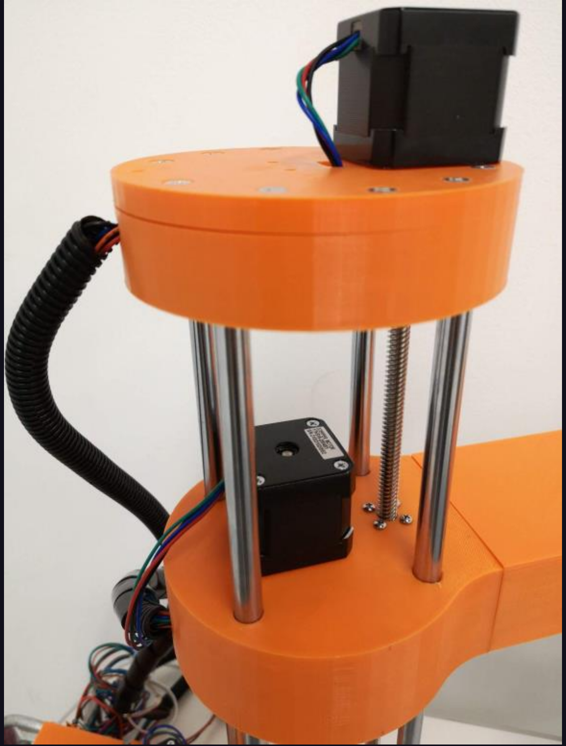
Digital Fabrication:

- Flashforge Creator Pro used for printing all non-standard components

Electronics and Robotics:

- Wiring, soldering, heat gun, and some special components used

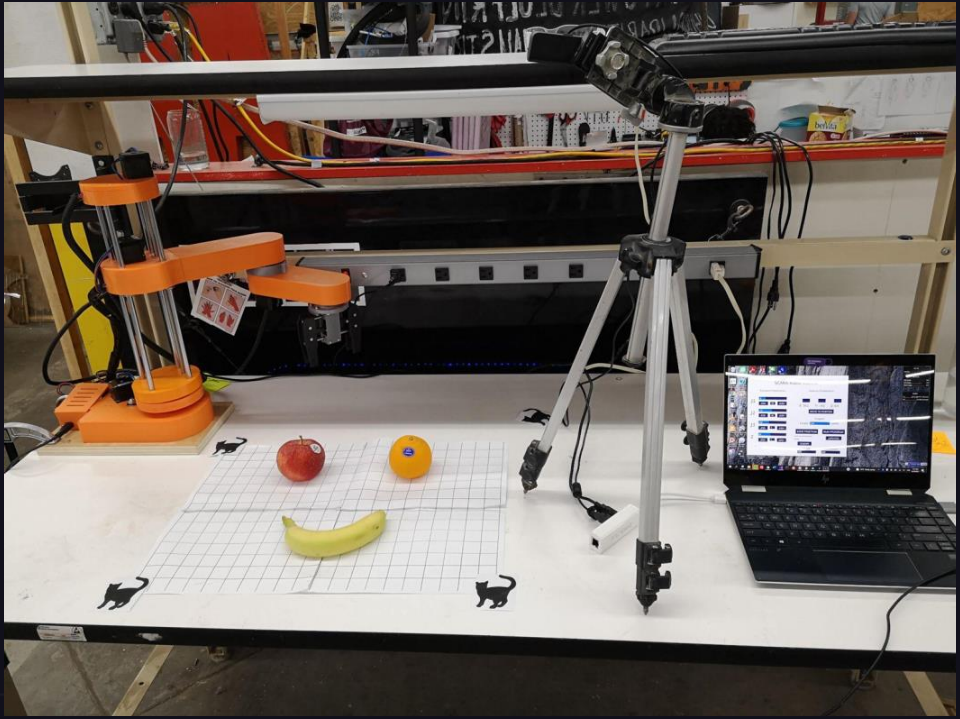
Z-Axis and Gripper



Stepper motor powers lead screw; robot slides on linear motion rods

Rotational motion of servo translated into linear “pinching” motion

Camera setup



Top Camera: Tripod is slightly in the frame, a flaw

Front Camera

03

ELECTRONICS INTERFACE

Linking the Computer Vision software with motor movement

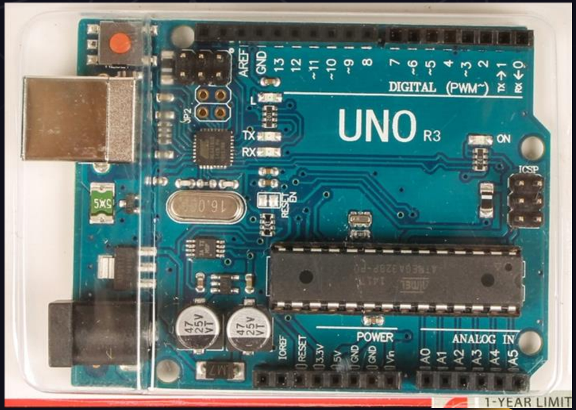
Arduino Control

Microprocessor: Inland Uno

Code is a combination of Arduino code and Processing code (for GUI)

Robot receives commands via Arduino Serial

Interface: Computer runs ML/computer vision Python script; Python sends commands to Arduino Serial; Arduino moves robot limbs




```
/*
 data[0] - SAVE button status
 data[1] - RUN button status
 data[2] - Joint 1 angle
 data[3] - Joint 2 angle
 data[4] - Joint 3 angle
 data[5] - Z position
 data[6] - Gripper value
 data[7] - Speed value
 data[8] - Acceleration value
*/
```

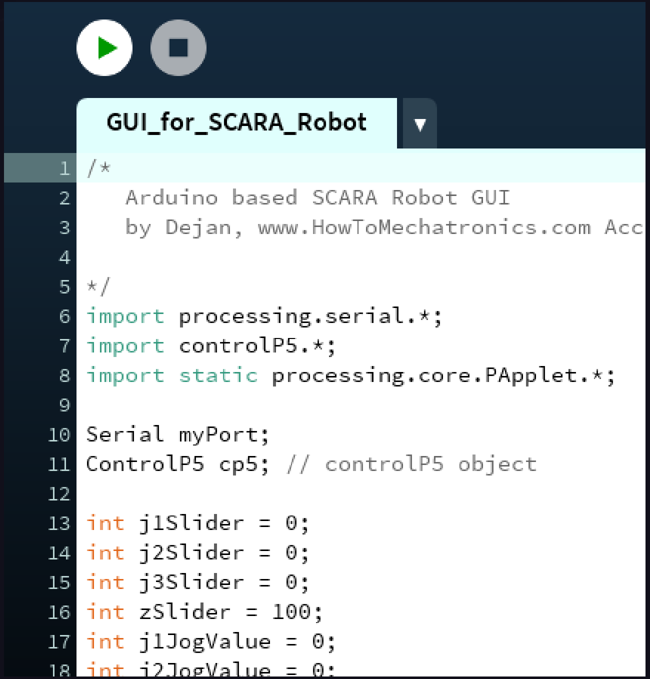
Arduino Code

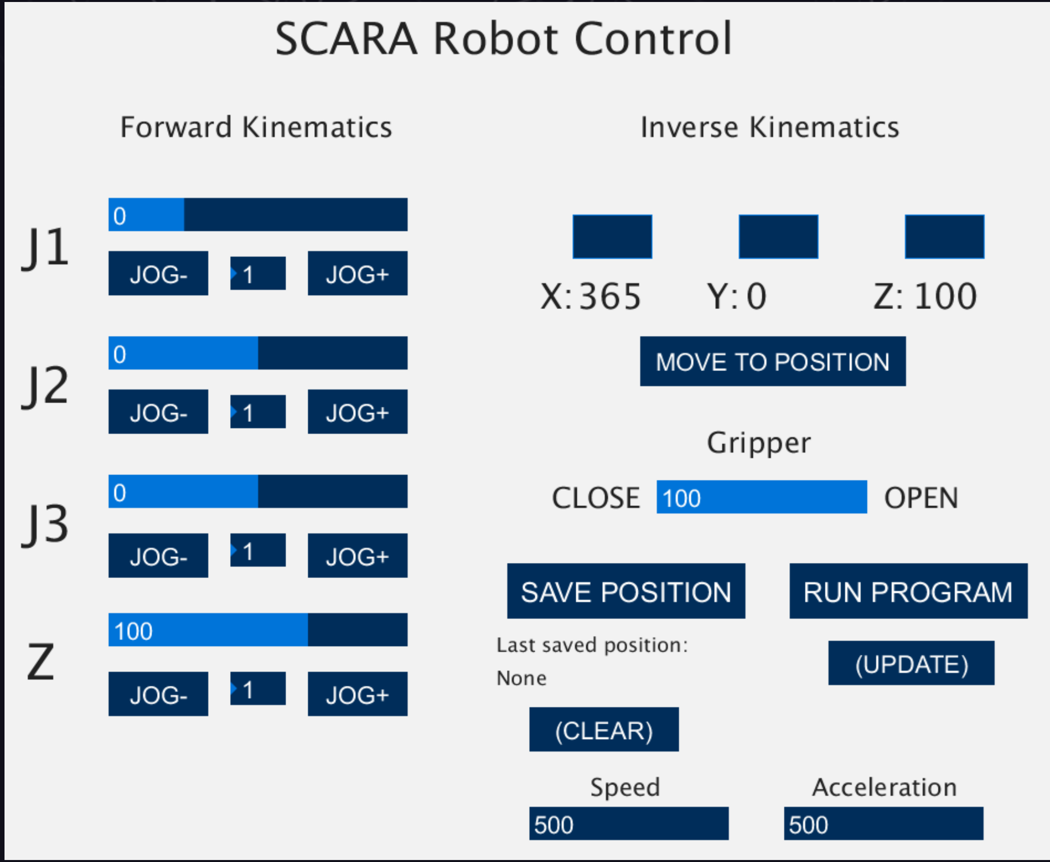
```
void loop() {

  if (Serial.available()) {
    content = Serial.readString(); // Read the incoming data from Processing
    // Extract the data from the string and put into separate integer variables (data[] array)
    for (int i = 0; i < 10; i++) {
      int index = content.inc
      data[i] = atol(content.
      content = content.subst
    }
    /*
```

```
void homing() {
  // Homing Stepper4
  while (digitalRead(limitSwitch4) != 1) {
    stepper4.setSpeed(1500);
    stepper4.runSpeed();
    stepper4.setCurrentPosition(17000); // When limit switch pressed s
  }
  delay(20);
  stepper4.moveTo(10000);
  while (stepper4.currentPosition() != 10000) {
    stepper4.run();
  }
```

Credits to Dejan from HowToMechatronics for the Arduino and Processing code, as well as the robot schematics

Processing GUI





The default GUI provided with this SCARA arm instruction guide. Will be modified to allow user to input the type of an object to pick up.

Forward/Inverse Kinematics

```
// FORWARD KINEMATICS
void forwardKinematics() {
  float theta1F = theta1 * PI / 180;   // degrees to radians
  float theta2F = theta2 * PI / 180;
  xP = round(L1 * cos(theta1F) + L2 * cos(theta1F + theta2F));
  yP = round(L1 * sin(theta1F) + L2 * sin(theta1F + theta2F));
}
```

Executed within the code for the GUI

I.K.: user chooses values of (x,y) for the arm to move to; code calculates the joint angles needed to move the arm to those coords

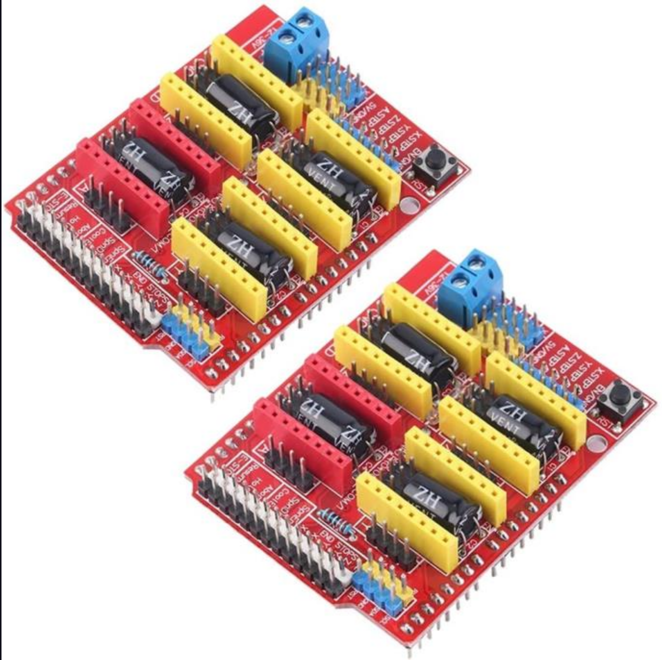
```
// INVERSE KINEMATICS
void inverseKinematics(float x, float y) {
  theta2 = acos((sq(x) + sq(y) - sq(L1) - sq(L2)) / (2 * L1 * L2));
  if (x < 0 & y < 0) {
    theta2 = (-1) * theta2;
  }

  theta1 = atan(x / y) - atan((L2 * sin(theta2)) / (L1 + L2 * cos(theta2)));

  theta2 = (-1) * theta2 * 180 / PI;
  theta1 = theta1 * 180 / PI;

 // Angles adjustment depending in which quadrant the final tool coordinate x,y is
  if (x >= 0 & y >= 0) {       // 1st quadrant
    theta1 = 90 - theta1;
```

Stepper Motors + Drivers

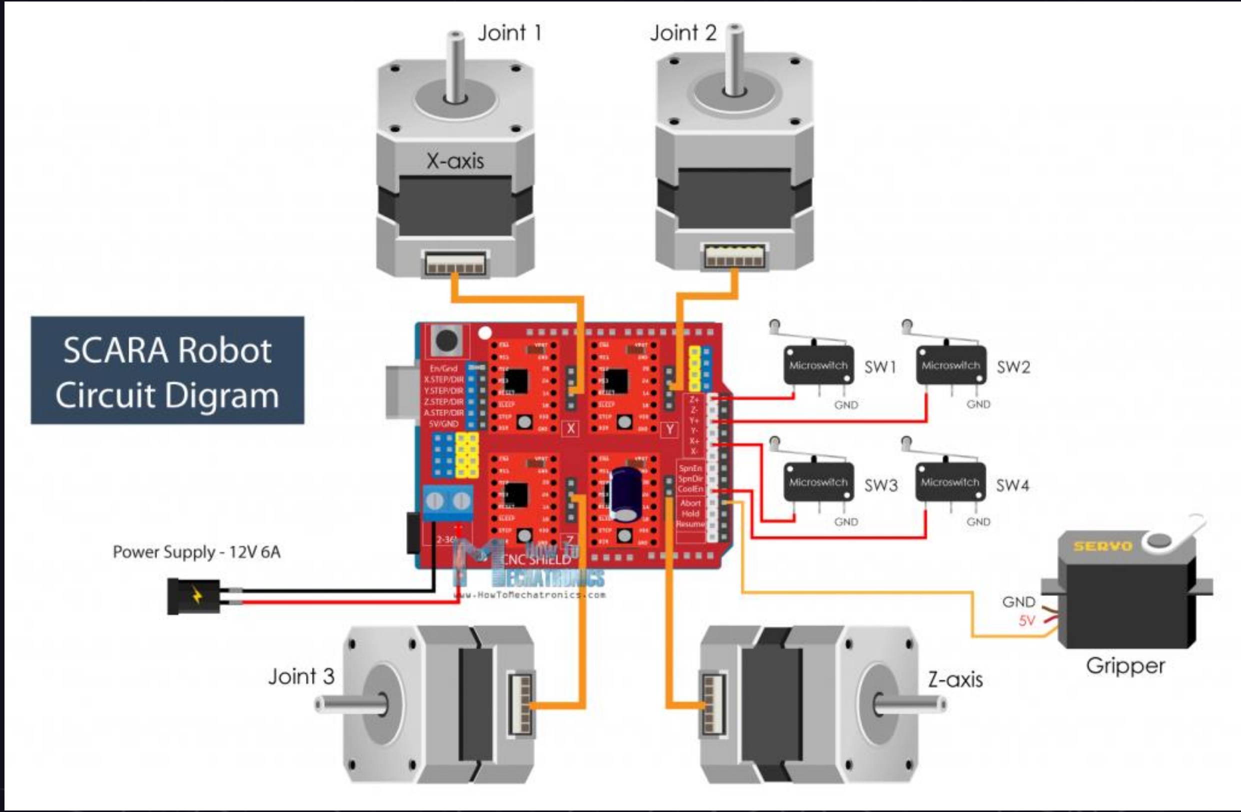
J1/J2/J3/Z: powered by NEMA 17 stepper motors

CNC shield used to supply 12V power to four stepper motors

A4988 Stepper Driver sits on top of capacitors to control steppers



CNC Shield Schematic



Current Electronics Debug

CNC Shield can power 4 motors: ”X/Y/Z/A” (A for special function)

J1/J2/J3/Z-axis in this project

Pins 2-7 are working as intended, but 12 + 13 are fussy on the Inland

J1, J2, J3 work; Z-axis and the gripper still need to be debugged

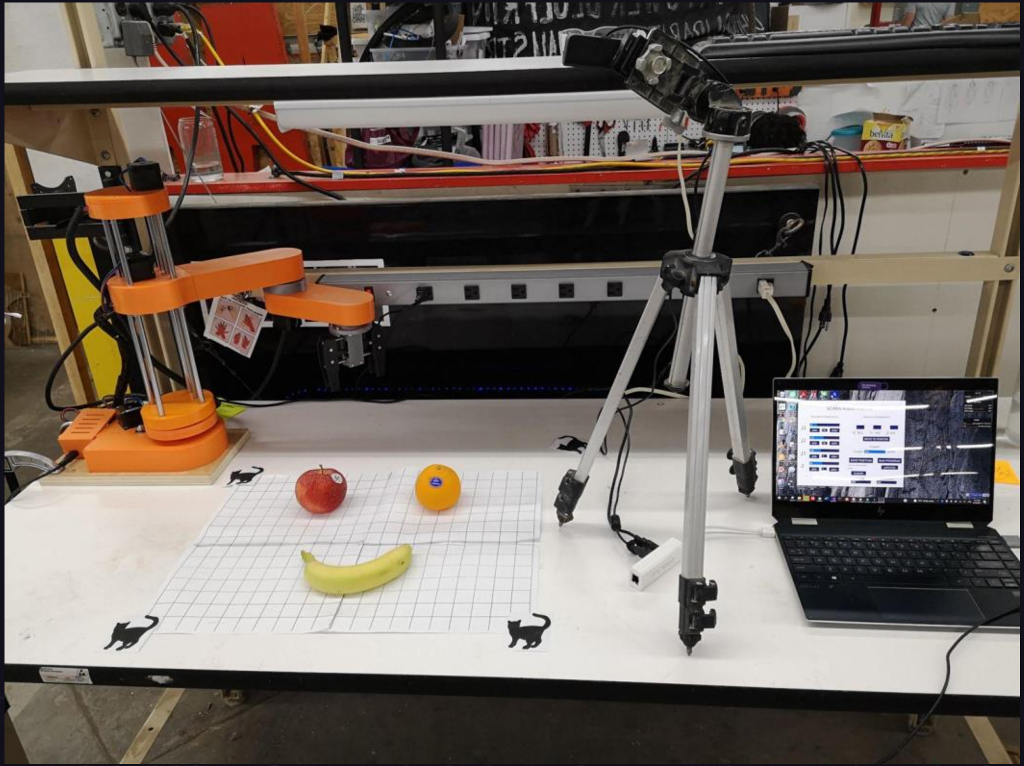
```
#include <AccelStepper.h>
#include <Servo.h>
#include <math.h>

#define limitSwitch1 11
#define limitSwitch2 10
#define limitSwitch3 9
#define limitSwitch4 A3

// Define the stepper motors and the pins the will use
AccelStepper stepper1(1, 2, 5); // (Type:driver, STEP, DIR)
AccelStepper stepper2(1, 3, 6);
AccelStepper stepper3(1, 4, 7);
AccelStepper stepper4(1, 12, 13);
```

04

NEXT STEPS

What’s left of the current project, and how we hope to expand our robot’s capabilities

Current Status



Necessary fixes and goals

- Z-axis and gripper need to be debugged
- Debug Inverse Kinematics in the GUI
- Physical apparatus needs to be designed to suspend camera over grid without a tripod getting in the frame
- Computer vision code needs to be applied to two cameras at once.
  - Technologically, not difficult: MobileNet SSD doesn’t take up much CPU per camera
- Grip strength of gripper needs to be tested with apple, orange, banana
  - If gripper performs poorly, new gripper hands will be designed
- Design a custom GUI that allows user to type in a classification of object to pick up

Suggested / Wish List

- Design a new E-box that can fit extra wires
- Place robot arm farther onto grid--current range of arm for picking up objects is limited
- Change the listed classification for “cat” and “bird” to “corner” for better appearance
- Implement voice commands so user can say “Pick up the apple”
- Add smart gripping functionality, so the gripper holds squishable objects (ex. banana) with a softer grip
- Improve the Serial method of commands: sending two commands at once causes undefined behavior

References

CV-Tricks. “Zero to Hero: Guide to Object Detection Using Deep LEARNING: Faster R-Cnn, Yolo, Ssd.” 28 Dec. 2017, cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/.

Liu, Wei et. al. Cornell University. “SSD: Single Shot MultiBox Detector.” 8 Dec 2015.

Nedelkovski, Dejan. How-To Mechatronics. “How to build your own Arduino-based robot.” Web. https://howtomechatronics.com/projects/scara-robot-how-to-build-your-own-arduino-based-robot/

Q&A