Generalized Robotic Injection

Joe Toth (67469069)
Derick Vasquez (79044774)
Trevor Stuart (91073470)
Alex Mason (13783783)

MECH 453 Fall 2024
9/20/2024

Abstract:

The following report details modifying an existing robotic arm kit into a device capable of performing automated injection, integrating an ultrasonic distance sensor, IP camera, and computer vision for real-time feedback of this procedure. The kit-provided end effector was replaced with a 3D-printed housing containing the above sensors and a makeshift syringe to simulate the injection. MATLAB was utilized to produce a control applet that takes incoming data over serial for the ultrasonic sensor and HTTPS for the camera. The IP camera and MATLAB's Computer Vision Toolbox provide video feed and analysis, allowing the robotic arm to locate an injection site, whereas the ultrasonic sensor provides depth perception. From the incoming data, the control software locates an injection patch (represented by a QR code), determines the injection depth, creates the injection trajectory, and proceeds with the injection. This process may be repeated automatically across the robot's usable workspace. This system has potential applications in the medical industry, replacing a tedious process with an automated solution. Leveraging 3D printing techniques, the premade robotic arm, and several programming libraries, this project has demonstrated a proof of concept for affordable automated robotic injection. Details of the design, integration, and programming are provided, allowing future students and engineers to replicate and improve upon this work.

1. Introduction

Most injections require skilled labor to administer. The developing world lacks the physical and human capital to immunize the population adequately. In the developed world, this labor can be devoted to patient care or other areas where more utility might be gained. Thus, there exists a need for a cost-effective generalized robotic injection requiring little to no skilled labor to operate. Our project goals are twofold:

1. Integrate a computer vision system that will aid in automating the injection process.

2. Administer injections using a three degree of freedom robotic manipulator automatically.

Nomenclature:

- PWM: Pulse width modulation

2. Methods

2.1. Sensor, Trajectory, and Control Software Overview

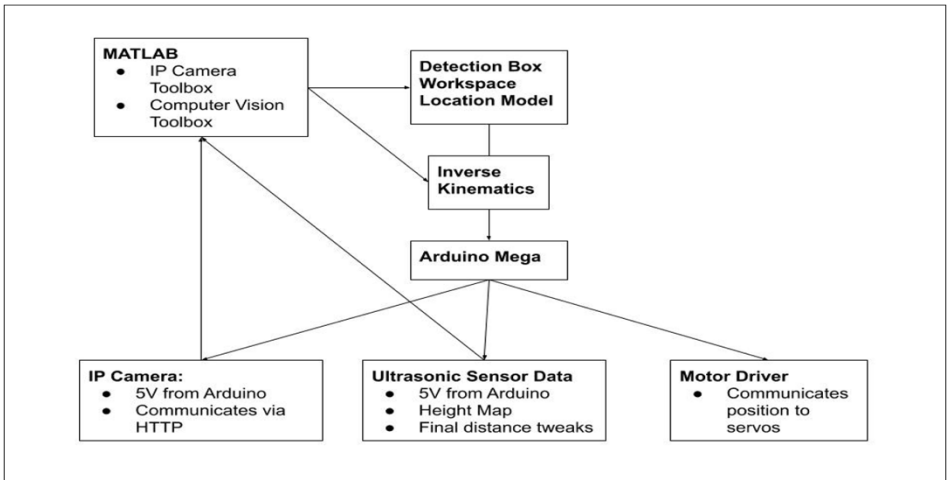

Figure 1: Robot Data and Control Workflow

The control overview in Figure 1 provides a detailed representation of the data and power flow between the various components within our robotic injection system. The system's heart is the MATLAB programming software, which is the primary computational engine for inverse kinematics calculations. These computations use data acquired from two sensing sources: the ultrasonic sensor and the IP camera. By leveraging the IP Camera Toolbox and the Computer Vision Toolbox within MATLAB, we can identify the target, utilizing the detection box as the primary locating feature to determine positional information. The Arduino serves two roles in the system. It functions as the central controller for the communication between components while supplying power to the ultrasonic sensor and the IP camera. Once MATLAB processes the necessary inverse kinematics solutions, the computed joint angles are transmitted to the Arduino through serial communication. Then, the Arduino relays these commands to the motor drivers, which control the servo motors. The entire process operates continuously and in real-time during the seek-and-injection operation, enabling seamless coordination between sensing, computation, and actuation.

2.2. Robot Components and Construction

2.2.1. Components

Figure 2: 6-DOF Robotic Arm Kit[1]

Figure 2 depicts a robotic arm kit like the one used in this report. This kit, or one similar, will provide the foundation and sufficient servos to replicate the injection robot.

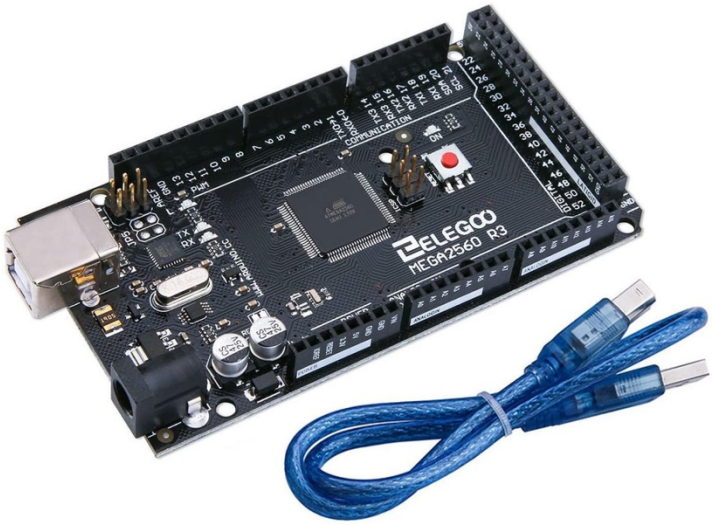

Figure 3: Arduino MEGA 2560 or Equivalent[2]

An Arduino Mega 2560 (or replica from 3rd party) microcontroller with USB acts as the communication between the control applet, sensors, and servo shield. An example is provided in Figure 3.

[1] (Yanmis, 2024)
[2] (Elegoo, 2024)

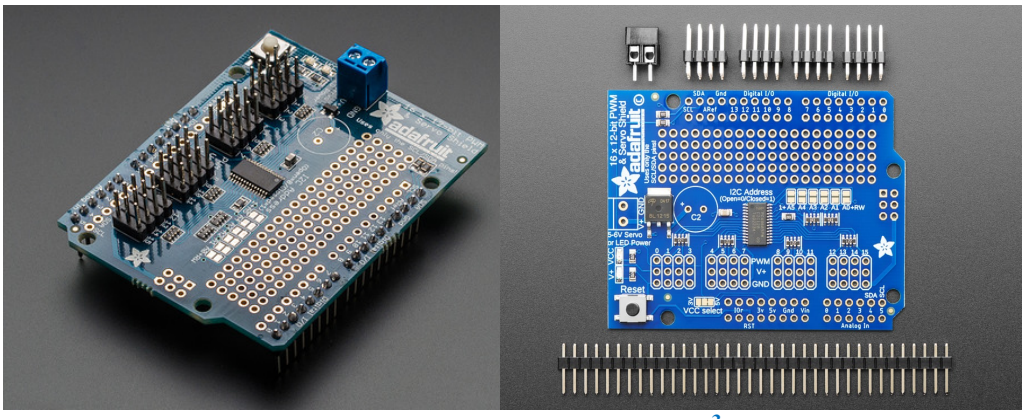

Figure 4: Adafruit PWM Servo Shield[3]

Figure 4 shows the servo shield used to control the robotic arm's movement. This specific board is necessary as it employs the Arduino library discussed later.

Figure 5: ESP32-IPCamera (left) and HC-SR04 Ultrasonic Transducer (right)[4,5]

The sensors used to locate the injection site are an esp32-ipcamera and a HC-SR04 ultrasonic distance sensor, both pictured in Figure 5. The HC-SR04 communicates and takes 5V from the microcontroller whereas the esp32-ipcamera only takes power.

Not Pictured:

- 5V DC Power Adapter
- Micro-USB Power Cable
- Male-Female and Male-Male Jumper Wires
- Electrical Tape
- Dry Erase Marker

Necessary Tools:

- Wire Splitter and Cutter
- Socket Set or Adjustable Crescent Wrench
- Flat and Regular Head Screwdriver for Small Screws
- Access to 3D Printer

[3] (Adafruit, 2024)
[4] (ai-thinker, 2024)
[5] (ITead Studio, 2023)

2.2.2. *Sensor Housing and Mount*

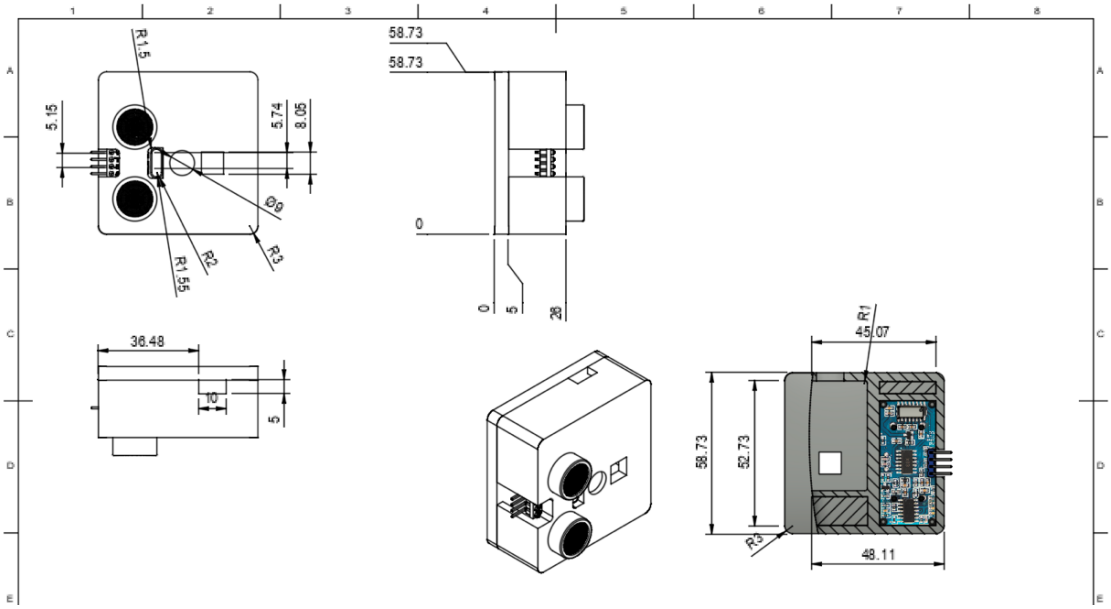

Figure 6: Technical Drawing of Sensor Housing

Figure 6 illustrates the housing/end-effector used to accommodate the sensors and makeshift needle. On the sides of the wider shell, two cutouts provide access to the pins of the HC-SR04 and USB of the esp32-ipcamera. The face of the housing has additional cutouts for the HC-SR04 transducers/oscillator and the camera. Away from the transducers, the back plate indexes to the shell with two slots. Through-holes can be drilled on the back plate for self-tapping screws in whatever configuration is necessary to accommodate various robotic arm kits.

2.2.3. *Bill of Materials*

Bill of Materials	
Item	**Cost**
Robot Arm Kit	$60[1]
Mega 2560 Controller	$20[2]
Adafruit Servo Shield	$17.50[3]
ESP32-IPCamera	$13[4]
HC-SR04 Ultrasonic Sensor	$3[5]
5V DC Power Adapter	~$6
Micro-USB Power Cable	~$3
Jumper Wires	~$10
Electrical Tape	~$7
Dry Erase Marker	~$1
Total:	**~$140.50**

Table 1: Bill of Materials for Project

Table 1 tabulates the cost of relevant components for the injection robot. Most of the cost is from the robot kit and electronics. Note that the cost of necessary tools, including access and 3D printing, is not included.

2.2.4. Assembly Process

The following lists provides a general process for integrating the sensors, housing, and robotic arm kit to arrive at the robotic injector:

1. Once printed using PLA or Petg, the ultrasonic sensor and IP camera are placed into the housing as show in Figure 6.

2. Remove the provided end-effector from the robotic arm kit and reduce the DOF to 3 (remove both wrist joints as well as the clamp). From the resulting arm, measure and drill the holes necessary on the back plate of the sensor housing.

3. Attach the sensor housing to the arm and securely locate the Arduino MEGA 2560 somewhere close to the base of the robot.

4. A dry erase marker should be cut down to minimal 30mm-40mm in length and installed into the socket on the front of the sensor housing.

5. Solder the headers/power terminal onto the servo shield, if necessary, before placing it into the Arduino MEGA. Be sure that the SCL and SDA pins align between the controller and shield.

6. Strip the barrel-side of the 5V power adapter before placing the negative and positive leads into the servo shield's power terminal. (Avoid powering anything on at this point)

7. Strip the regular USB side of the micro-USB power cord before connecting the leads to jumpers. Subsequently, place the positive and negative jumpers into a 5V and GND pin on the Arduino.

8. Connect jumpers from the ultrasonic sensor's pins to the Arduino. Anchor all wiring to the frame on either side of all pivots with electrical tape or wire ties.

9. Connect the Arduino to a computer via USB and upload the Arduino control software. Provide power to the servo shield

10. Open the MATLAB control applet and test the connection via the motor configurator.

Refer to the following section for a more detailed look at the wiring process between the controller, servo shield, motors, and sensors.

2.3. Wiring Diagram

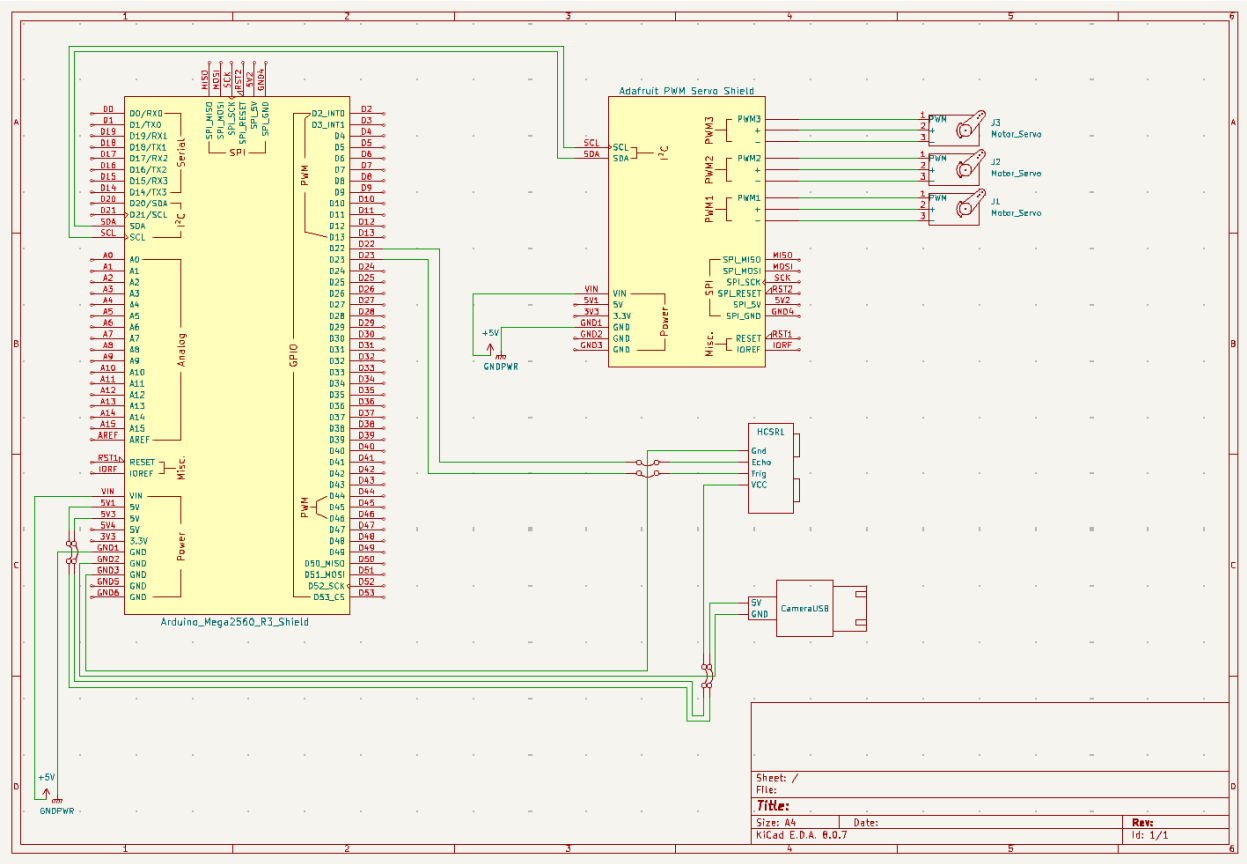

Figure 7: Component Wiring Diagram for Injection Robot

The robot's electric wiring is in Figure 7. The system requires two power supplies to operate; the first is supplied from a USB connection to the Arduino MEGA, and the second from a 5V DC adapter to the servo shield. Power for the esp32-IP camera is sourced from the 5 VDC bus located on Arduino MEGA. The HC-SR04 ultrasonic VSS and ground are connected to 5 VDC buses on the Arduino MEGA. Echo terminal shall be terminated on terminal D22, and Trig shall be terminated on D23. The Adafruit PWM Servo Shield requires a 5 VDC supply from the DC bus. The connection between the Mega's and Servo Shield's SCL and SCA terminals is essential for the proper operation of the robot. The servos J1, J2, and J3 should be terminated in the same order as PWM1, PWM2, and PWM3. The following section goes into greater detail on the MATLAB-side control software which utilizes the above electrical connections.

2.4. MATLAB Control Software and Applet

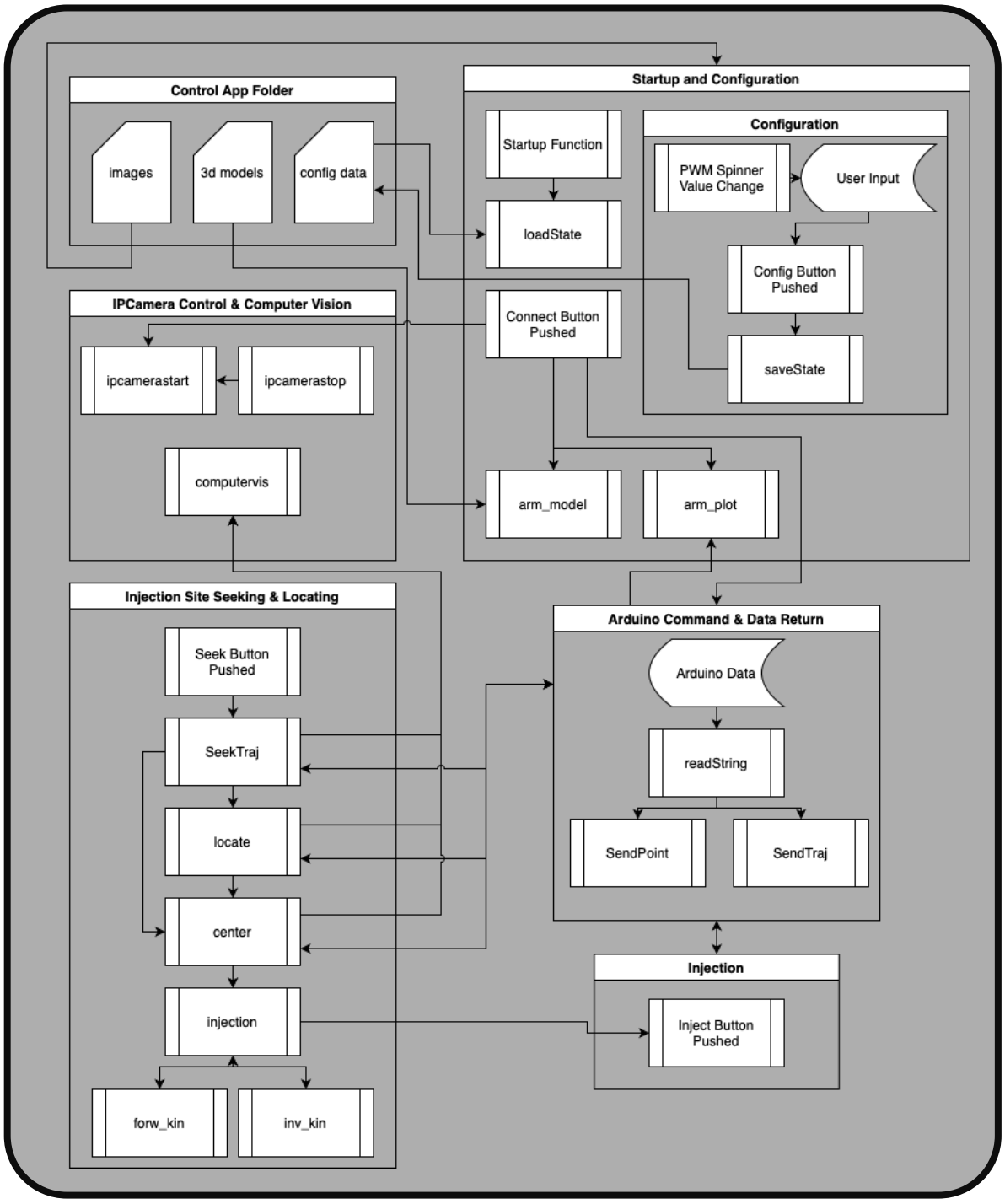

Figure 8: MATLAB Applet's Control Workflow Diagram

Figure 8 depicts a simplified version of the MATLAB control applet's workflow. A main folder contains the .mlapp file, sub-directories for images and 3D models, and a user configuration data file. Upon opening the app, the Startup Function attempts to load previous user configuration data; if there is no data present, the app will set the relevant variables to their defaults before displaying the main control tab GUI. If the secondary configuration tab changes user

configuration data or calibrates the PWM ranges, a *saveState* function updates the configuration data file upon request.

The MATLAB control software uses two main functions for dictating movement: *SendPoint* and *SendTraj*. The *SendPoint* function receives joint angles for joints 1-3 in degrees and turns them into PWM by interpolating the angles and user-provided configuration data. Then, the PWM values are combined into a single string headed by the desired command type. This header tells Arduino whether to return ultrasonic distance data upon arrival at the desired location. Similarly, the *SendTraj* function receives a matrix of waypoints and a time interval between commands. A type header is also included to determine whether data should be sent back with each point.

When the *Connect* button is pushed, the software connects to the Arduino, creates the arm model and configuration plots, initializes the IP camera, and finally sends the robot to its home configuration. The user is then presented with the status of the Arduino connection, the robot's position, and the ability to disconnect from the serial port. Two additional buttons (*Seek* and *Go Home*) are at the bottom of the control tab. The *Go Home* button sends the robot back to its home configuration.

If the *Seek* button is selected, the *SeekTraj* function is called, which sends a hard-coded seeking trajectory to the *SendTraj* function along with the command to send distance data back. Within the *SendTraj* function, a moving average of the acquired distance data is computed, and the robot is told to stop once the average drops a few centimeters from the previous window. Then, the *computervis* function tries to locate a QR code from a snapshot of the live video feed. If the injection site is located, the *SeekTraj* function continues. However, the locate function is called if *computervis* cannot locate the QR code. Here, the robot changes the angle of joint 1 three times using the *SendPoint* function. From the resulting distance data, the software chooses a direction to start turning until the injection site is located. Once a detection box is drawn by *computervis*, the *center* function is called.

The *center* function shifts the perspective of the end-effector until the detection box is centered within the live stream video feed. Once the detection box is centered, the injection function is called. Here, the robot shifts joint angle 1 so that the ultrasonic distance sensor can take readings over the injection site. The resulting data is then used to compute an injection trajectory, considering the radial distances from the sensors to the injection marker. A new UI panel is subsequently created containing a height map from the seek trajectories distance data, an *Inject* button, and a *Cancel* button.

If the user pushes the *Inject* button, the robot sends the injection trajectory to the *SendTraj* function and proceeds with the injection. The control software allows for the placement of the injection site onto objects of varying height and location within the workspace. If it is within reach, the software will repeatedly and reliably automatically sense the injection site and compute the injection trajectory.

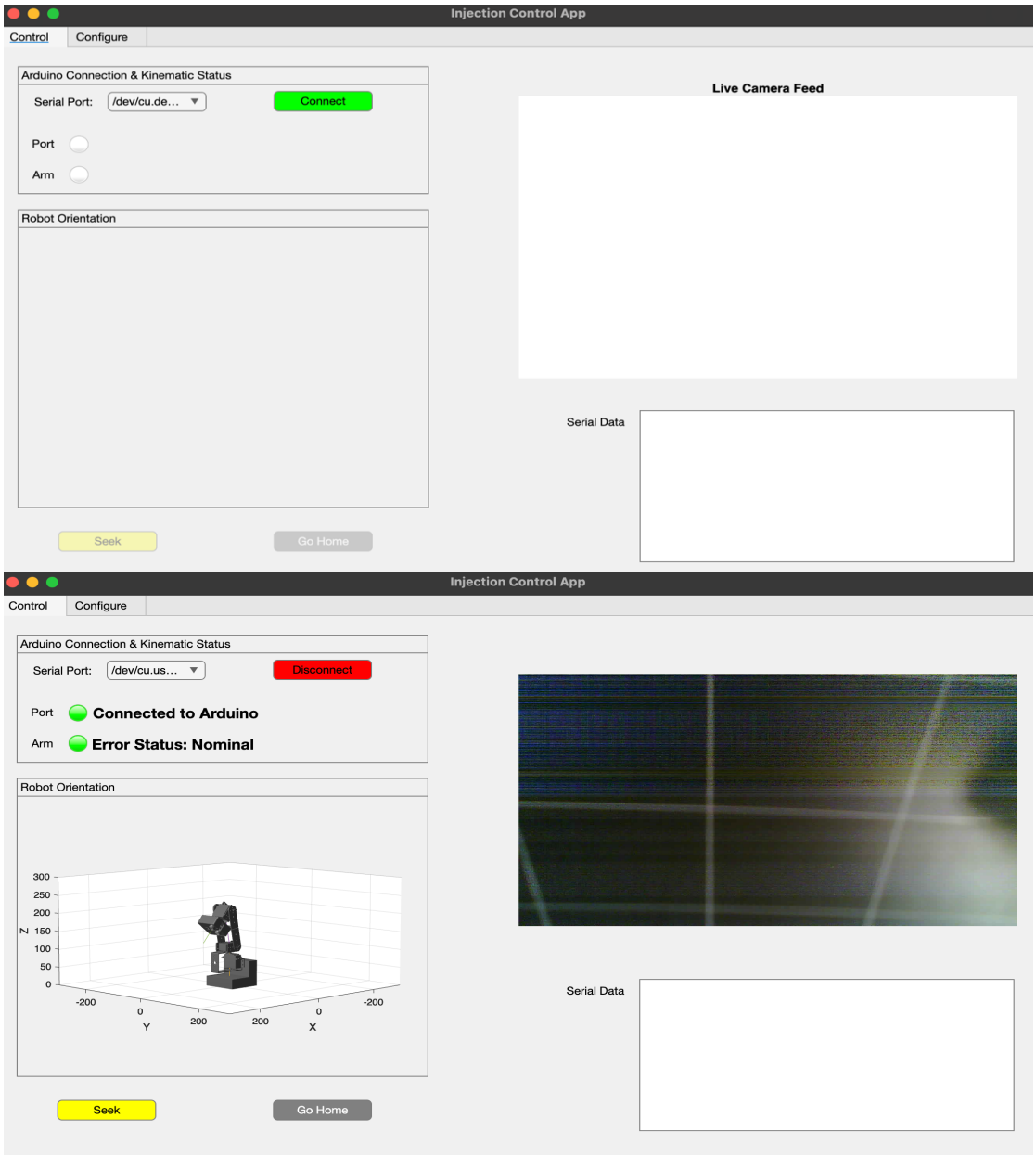

Figure 9: MATLAB Applet's Control Tab

Figure 9 displays the primary GUI for the MATLAB control applet. The correct serial port is selected from this window, and the Connect button is pushed; then, the camera feed and the robot's orientation are displayed. The user may now push the *Seek* button to actively look for the injection site or the *Go Home* button to return to the home configuration. A debug text box in the bottom right corner displays incoming data from the Arduino. The difference between the top and bottom frames represents the results of pushing the *Seek* button.

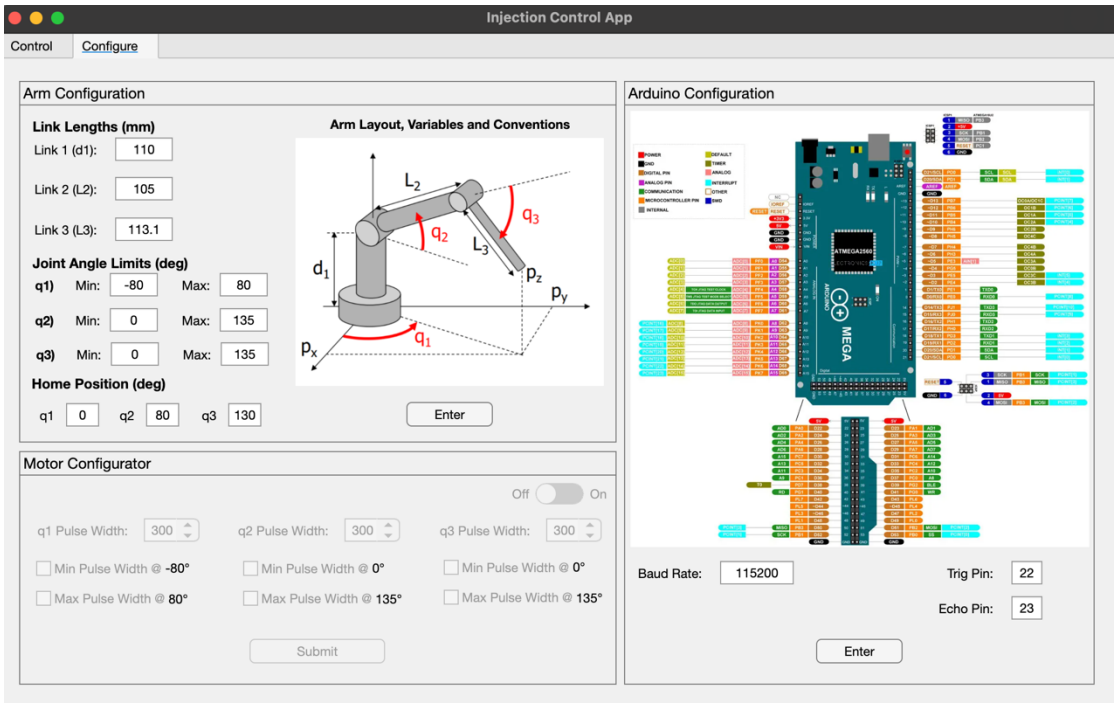

Figure 10: MATLAB Applet's Configure Tab

Figure 10 shows the secondary GUI, where the user sets link lengths, joint angle limits, and desired home position within the Arm Configuration container according to the depicted conventions. Similarly, in the Arduino Configuration container, the baud rate and the data pins for the ultrasonic transducer are set. The *Enter* button in each container updates the current user configuration data file, which will be loaded upon the app's next startup. Finally, the Motor Configurator container allows for the incremental calibration of PWM limits. Checking a box sets a value for the corresponding joint angle, and the *Submit* button updates the user's configuration data file.

2.5. Arduino Command Software

The Arduino command software utilizes the *Adafruit PWM Servo Driver*[6] and *HC-SR04*[7] Arduino IDE libraries to communicate with the servo shield and ultrasonics sensors, respectively. First, objects for both sensors are created, and then all variables are initialized. The serial baud rate, the servo shield controller's frequency, and the PWM frequency are set within the startup loop. Within the run loop, an if statement gates the following commands until serial data is received. Once the command string is received, the first entry and the rest of the array are parsed into *typeString* and *dataString* containers. The first entry determines the type of function requested from MATLAB, and the rest is the input data. Then, a switch block determines the appropriate function based on the value of *typeString*.

A first entry value of '1' denotes the configured command calibrating each motor's joint/PWM ranges. Here, the dataString is parsed into the joint number and the change in PWM for the associated motor. The first entry value of '2' represents a 'point command' where the input data

[6] (Gundry, 2024)
[7] (Šošić, 2022)

is parsed into three PWM values (one for each joint). Collectively, these PWM values send the end effector to a selected point. Finally, a first entry value of '3' follows the same process as with '2' but sends the current ultrasonic distance measurement back to MATLAB over serial. The C++ code for the Arduino IDE file is in Reference Figure 11.

3. Results and Discussion

Our scope for this project changed from the midterm report. Initially, we wanted the ability to switch injection angles between subcutaneous and intramuscular. However, the limited dexterity of a 3DOF robotic arm reduced our scope to automatically sensing and touching an injection site within the workspace. Through various prototyping and software development efforts, the result of this project has been the creation of a robot that can autonomously locate and move an end effector to a specific point, simulating the injection process. It does this through a series of steps consisting of generating a workspace around the injection's location, seeking and identifying the datum to pinpoint the injection location, and completing the injection. The robot can automatically detect the data over a variety of heights as well as locations in the workspace. For most heights, the injection works smoothly and repeatably. However, if the object is over 3 cm tall, the robot can locate the injection site and touch it, but the depth sensing becomes slightly impaired; the robot tries to go to a depth lower than the surface, putting some stress on the arm.

4. Conclusion

This robotics project has demonstrated significant development toward an affordable autonomous robotic injection system. The robot's capability to properly locate injection sites through integrating computer vision and robotic control provides substantial benefits for the medical industry. By modifying a relatively inexpensive robotic arm kit and utilizing software tools such as MATLAB, Arduino, and Computer Vision, the robot arm achieved its primary goal of integrating a computer vision system to locate injection sites and safely approach them with its end effector.

Although the second goal of administering intramuscular and subcutaneous injections has yet to be fully realized, the project has laid a strong foundation for future advancements. Additionally, the robotic arm could use a refined angle control system to be capable of both subcutaneous and intramuscular procedures and a collision detection system to further enhance its safety.

Overall, this project highlights the capabilities of low-cost, robotic solutions for tedious tasks in the medical industry. Integrating 3D printing, programming, and relatively inexpensive hardware has provided robust solutions and a foundational platform for further research and development. Future iterations could focus on enhancing the mechanical structure of the arm as well as its actuation capabilities. This would make significant progress towards achieving the full scope of the project. This project showcases the ability of practical robotic solutions for simple tasks like the common flu shot. From this project alone, we can see the potential of automated injection systems in healthcare applications.

Appendix

Reference Images

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <HCSR04.h>

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
UltraSonicDistanceSensor distanceSensor(22, 23);
#define SERVO_FREQ 50 // Analog servos run at ~50 Hz updates

String incomingString = "0";
int typeString = 0;
String dataString = "0";
int type = 0;
int pwmv = 0;
int q1 = 0;
int q2 = 0;
int q3 = 0;

void setup() {
  // Initialize serial communication
  Serial.begin(115200);
  // Initialize pwm communication
  pwm.begin();
  // Set onboard oscillator frequency
  pwm.setOscillatorFrequency(24000000);
  pwm.setPWMFreq(SERVO_FREQ);
  delay(10);
}

void loop() {
  if (Serial.available() > 0) {
    incomingString = Serial.readStringUntil('\n');
    typeString = incomingString.substring(0,1).toInt();
    dataString = incomingString.substring(1);
    switch (typeString) {
      // Configure Command
      case 1:
        type = dataString.substring(0,1).toInt();
        pwmv = dataString.substring(1,4).toInt();
        if (type==1){
          pwm.setPWM(0,0,pwmv);
        }
        if (type==2){
          pwm.setPWM(1,0,pwmv);
        }
        if (type==3){
          pwm.setPWM(2,0,pwmv);
        }
        break;

      // Point Command
      case 2:
        q1 = dataString.substring(0,3).toInt();
        q2 = dataString.substring(3,6).toInt();
        q3 = dataString.substring(6,9).toInt();
        pwm.setPWM(0,0,q1);
        pwm.setPWM(1,0,q2);
```

```
        pwm.setPWM(2,0,q3);
        break;

      // Point Command & Distance Send
      case 3:
        q1 = dataString.substring(0,3).toInt();
        q2 = dataString.substring(3,6).toInt();
        q3 = dataString.substring(6,9).toInt();
        pwm.setPWM(0,0,q1);
        pwm.setPWM(1,0,q2);
        pwm.setPWM(2,0,q3);
        Serial.println(distanceSensor.measureDistanceCm());
        break;
    }
  }
}
```

Figure 11: Arduino-Side Control Code

References

Adafruit. (2024). *Adafruit 16-Channel 12-bit PWM/Servo Shield - I2C interface.* Retrieved from Adafruit: https://www.adafruit.com/product/1411

ai-thinker. (2024). *ESP32-CAM camera development board.* Retrieved from ai-thinker: https://docs.ai-thinker.com/en/esp32-cam

Šošić, M. (2022). arduino-lib-hc-sr04.

Elegoo. (2024). *MEGA 2560 R3 Board with USB Cable.* Retrieved from Elegoo: https://us.elegoo.com/products/elegoo-mega-2560-r3-board?srsltid=AfmBOorgKsXhY1xUM44jRj6yu8LZjpCUaTBBp2frSmDDpqeMO2NaBnAP

Gundry, T. (2024). Adafruit-PWM-Servo-Driver-Library.

ITead Studio. (2023). *Ultrasonic ranging module : HC-SR04.* Retrieved from Electro Schematics: https://www.electroschematics.com/wp-content/uploads/2013/07/HC-SR04-datasheet-version-2.pdf

Yanmis. (2024). *6DOF Mechanical Arm Claw Kit, Manipulator Industrial Robot Mechanical Arm Automatic Robot Parts.* Retrieved from Amazon: https://www.amazon.com/Mechanical-Manipulator-Industrial-Gripper-Automatic/dp/B081TDD3QG/ref=asc_df_B081TDD3QG?mcid=80973d0d0ab830dbacf88bb84dfe4411&tag=hyprod-20&linkCode=df0&hvadid=696924910050&hvpos=&hvnetw=g&hvrand=15893387694397619474&hvpone=&hvptwo=