Project Report

Mike Hennessy and Kennedy Necoechea

MECH 598: Electromechanical Engineering

Xiaodong Sun

Mechanical Engineering Department

Loyola Marymount University

Los Angeles, California

May 1, 2024

Table of Contents

Introduction

Background

The final iteration of the gantry system working with the Fanuc 200iC robot involved a well-coordinated sequence. When a cube was placed on the platform, the NEMA-17 stepper motor rotated clockwise, propelling the platform from its start position to its end position. The platform halted until the block was removed by the robot arm. Conversely, when no block was present on the platform, the stepper motor rotated counterclockwise. This signaled a retraction of the platform until a block was positioned on the platform or it reached the starting point of the shaft. The gantry system had an established home position, where once the block was placed in its slot, it proceeded to move away from home.

The motor's operations were directed by an Arduino Uno and L298N Motor Driver that received input from the Ultrasonic Sensor. Since the sensor indicated how far away an object is, it detected if the block was placed on the platform by encoding 9 cm or less. The built-in LED as well as an additional LED would light up when the platform was in motion, or they turn off when the kill switches are reached. The speed of the motor was set to be changed from 250 to 350 by a potentiometer, where 300 is ideal. Furthermore, the report delves into the created gantry system and its full capabilities and cohesiveness with the Fanuc 200iC.

Schematic

The following wire schematic (Figure 1) is representative of the wiring used for the final implementation. Noting that wokwi.com [1] does not have L298N Motor Driver or a Fanuc Robot, some connections are missing in the schematic. Digital inputs 6 and 7 were utilized for the robot's input and output wires.

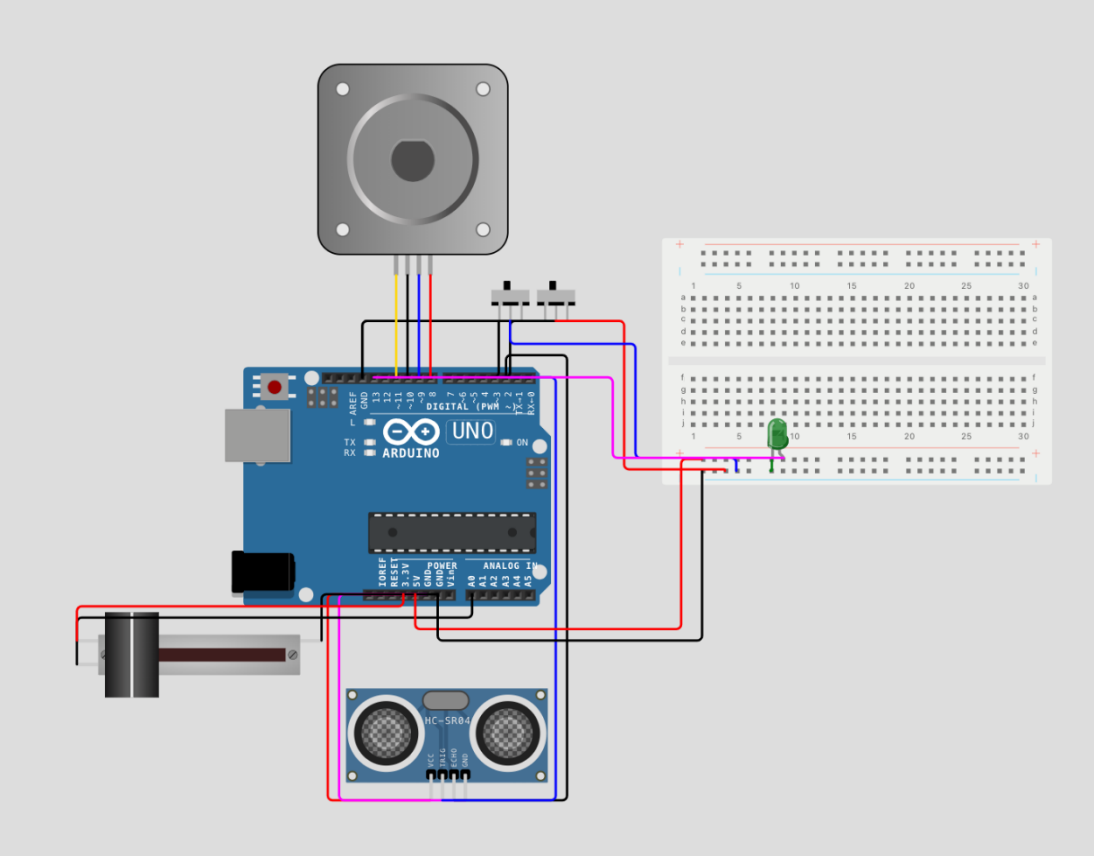

Figure 1: Wire schematic of the final implementation. [1]

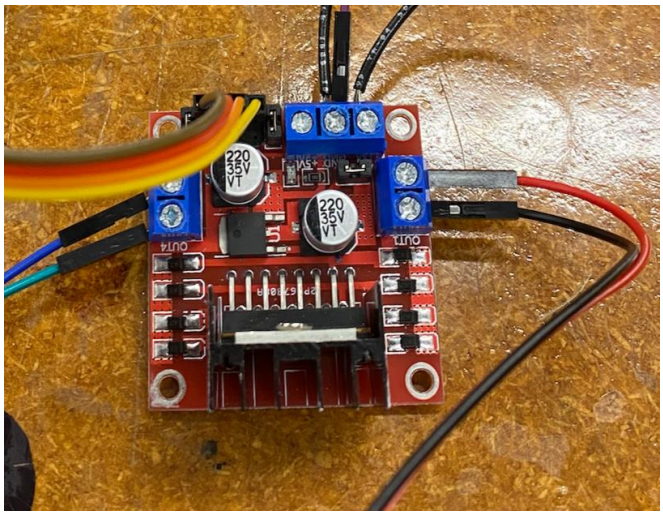

Figure 2: L298 Motor Driver connections.

The DC power supply's positive and ground connected to the corresponding terminals on the driver. It has two input power pins that accept 5-12V or 5-7V. The two screw terminals on the

left and right (Figure 2) are for each motor, motor A and motor B. The REMA-17 stepper motor was hooked up to these terminals for successful actuation.

I/O Assignment

Table 1: Arduino Uno pin assignment.

Part	*Pin*	*Mode*	*Description*
Potentiometer	A0	Analog Input	Controls stepper speed
Ultrasonic Sensor	2, 3	Digital Input	Senses block distance
Limit Switch	4, 5	Digital Input	Stops stepper rotation
Fanuc 200iC	6, 7	Digital Input	Robot's input and output connections
L298N	8, 9, 10, 11	Digital Output	Stepper motor connection to driver
Green LED	13	Digital Output LED Built-in	Platform is moving
Power Supply	V+	12V	L298N 12V Vin

Table 2: Fanuc Robot.

Function	*Description*
DO [111] ON	Robot is unlocked and ready to pick up/drop off block, LED is off, and Arduino is locked
DO [101] ON	Robot is at home position, LED is on, and Arduino is unlocked
DO [101] OFF	Robot is unlocked and ready to pick up/drop off block, LED is off, and Arduino is locked
DO [111] OFF	Robot is at home position, LED is on, and Arduino is unlocked

Flowchart

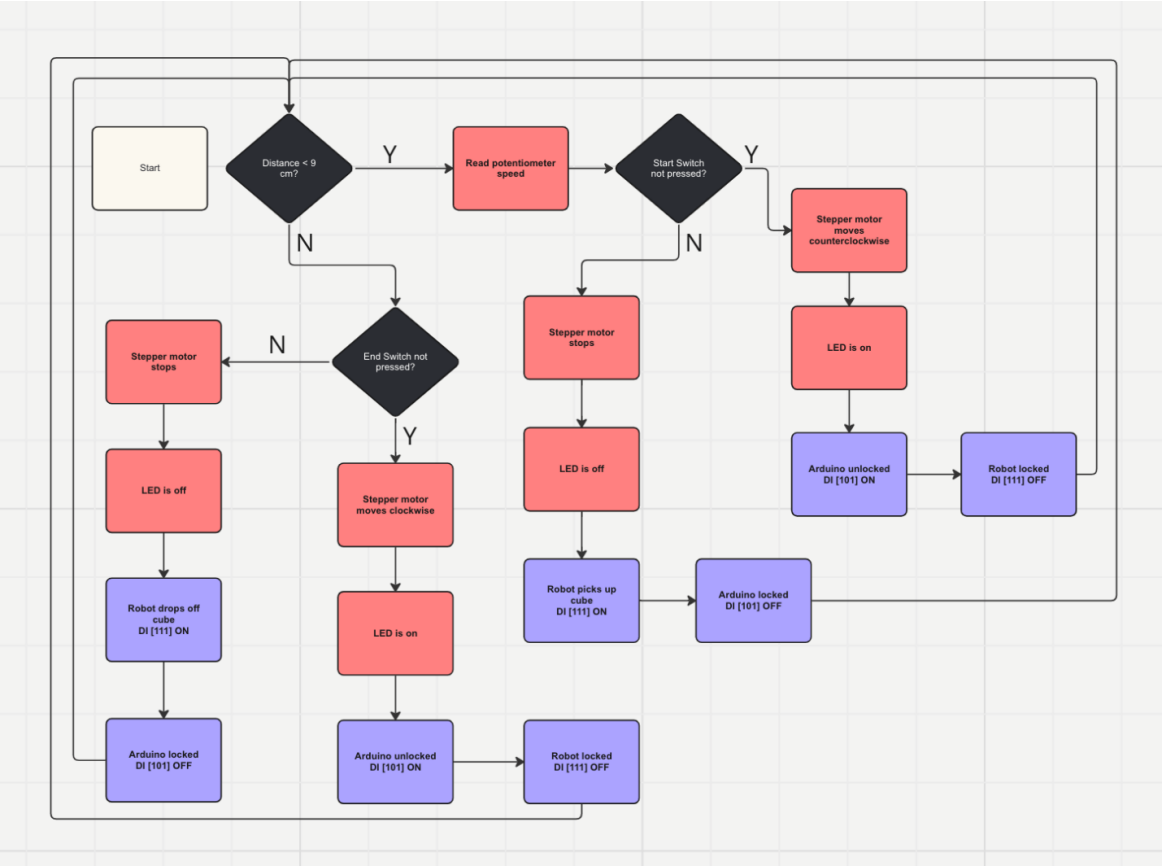

Figure 3: Flowchart of robot and Arduino code implementation. [2]

Simulation and Implementations

For the first part of the project, the Arduino code was written simultaneously as the hardware was built. The Arduino Uno code and breadboard were simulated using Wokwi. The gantry was built to make a movable platform on two shafts, controlled by a stepper motor. As the stepper motor turned clockwise and counterclockwise, the platform moved forward and backward, respectively. After achieving a functioning simulation, the physical hardware was built, and the code from Wokwi was put into an Arduino file for testing. Since the Wokwi simulation has great accuracy, not many changes were made between the simulation and the physical adaptation. One change that was necessary was adding the limits switches because they were not on Wokwi's materials list. After completing the Arduino, the project had multiple functions fitting the requirements:

- An ultrasonic sensor sensed if the 1 by 1 by 1 inch block was on the platform by checking the distance of the nearest object.
- An LED turned on when the platform was in motion with the block on it.
- A potentiometer could be manually turned to control the speed of the platform.
- Limit switches at the beginning and end of the shafts set the speed of the stepper motor to zero after being clicked.

When the block was placed on the platform, the stepper motor moved clockwise to push the platform forward until the block was taken off the platform or it reached the end of the shaft. When the block was not on the platform, the stepper motor moved counter-clockwise to push the platform backward until the block was placed onto the platform or it reached the beginning of the shaft.

After completing the Arduino aspect of the project, we moved onto the robot. ROBOGUIDE software was used to simulate the motion and functions of the FANUC 200iC robot. A SOLIDWORKS CAD model representing the gantry system was placed virtually in ROBOGUIDE for a precise simulation. From there, we set five locations. The first was the home location. This is where the robot started and began its functions. After, we added locations A and B, where the robot would drop and pick up the block, respectively. The last two locations were slightly above positions A and B, which allowed the robot to have better positioning before dropping and picking up the block. After the positions were set, we programmed the motion with the following steps:

1. Begin at home position.
2. Move above position B.
3. Move to position B and pick up the block.
4. Move back to the home position.
5. Move above position A.
6. Move to position A and drop the block.
7. Move back to the home position.

The full motion steps can be seen in the *Source Code* section. To incorporate the two aspects of the project together, the Arduino called the FANUC robot's motion when the block reached the end of the shaft, where the block was picked up at position B. The platform then moved to the front of the shaft, and the FANUC robot was called to drop the block at position A.

Troubleshooting and Discussion

The design had a lot of problems throughout the semester, so a lot of troubleshooting was necessary to complete the project. Slight changes were made between the online simulation of the project and the physical assembly because there were differences in the virtual materials in Wokwi and the physical materials we had access to. The first troubleshooting came when testing the ultrasonic censor for accuracy. It was regularly accurate, but it was inconsistent between uses, and therefore, the distance from the sensor to the block that was used in the Arduino code was a wider range than initially intended. The second sensor that needed testing was the limit switches, otherwise known as the kill switches. We used two of them, and they worked well to start. However, with prolonged use, the switches gradually deteriorated, becoming increasingly prone to faults. By the time of testing, the switch at the home position worked less than half the time, which was detrimental to our project. Thus, we had to replace the hardware before attempting to capture our final video. Thirdly, we had problems with the adhesion between the shaft supports and the surface. The surface was 3D printed and made of PLA, and we tried using multiple advanced glues to attach the shaft supports to it, but those did not work. We then used hot glue, and that worked temporarily, but came apart multiple times and we had to redo the adhesion. This also brought alignment issues to the system, where the motor would not do a full revolution because the alignment was off. In hindsight, screws should have been placed between the parts to establish strong attachment. Lastly, the stepper motor ran into problems. It would not finish revolutions and then would go backwards. This was fixed by stabilizing it and making sure it was aligned properly. Further, its connection to the L298N motor driver had to be double checked and tinkered to get the desired output. The ENB pin was not in the driver, and that

limited the system's rotation because the clockwise rotation could not be activated. After inserting this pin, both the clockwise and counter-clockwise rotation functioned properly.

Aside from the hardware, the software also needed troubleshooting. After finalizing the flowchart, converting that flowchart into Arduino Uno code took both scripting and testing. Each time we ran the machine, we fixed a bug. After the bugs were all fixed, the machine ran smoothly, but that took many trials. Specifically, we had to test the speed of the stepper to make it a range that the motor could handle without malfunctioning. After both testing and doing online research on stepper motors, we found that a range between 250 and 350 on the potentiometer was most manageable [3]. The second software troubleshooting we did was with the limit switches. The literature on limit switches online was limited, so it took a lot of testing on how to use them properly [4]. This includes the script necessary to read whether the switches were clicked, which prongs of the limit switches must be connected to the ground, and which prongs must be connected to a pin.

The FANUC Robot gave us problems towards the end of the project. We simulated the entirety of the robot's movements in ROBOGUIDE, then the process was changed slightly when implemented with the physical product. The figures in the *Source Code* section show the finalized code. The problems and troubleshooting came when calling the digital output [DO] and digital input [DI] steps. The digital input did not turn off when it was supposed to, and therefore, the platform could not move. This happened because the DI was triggered in the Arduino code when the platform was at either position A or B. The bug was fixed by adding an extra Wait step of 1 second after turning on the DO to give time for the DI to turn off.

Future Work

If this project continued, firstly, the hardware would be fixed. An apparatus would be designed to fit every part of the physical project, including a slot for a bread board, the Arduino Uno, and the motor. Currently, only the shaft supports have a designed platform. Having an entire platform makes testing and modifications easier, has better presentation, and makes it more understandable for other viewers. Also, the platform would minimize bugs because many of the bugs we came across were machining and hardware issues that happened because of instability of the machine. Secondly, another improvement to the hardware would be soldering the wires into the breadboard and Arduino holes. The wires frequently detached from both the breadboard and the Arduino, so having a firm connection could limit the bugs. This could only be done after 100% accuracy was achieved because it is permanent.

Aside from organizing the hardware, there would also be software additions in the future. Currently, the FANUC robot simply drops a block and picks it up. With more time, we would advance the robot's functions. For example, instead of having one pick-up and one drop-off location, there would be multiple. The robot could pick-up and drop-off from both sides and from the middle after adding more locations and functions in ROBOGUIDE. Further, to make the project more relevant to real world applications, the robot could sense where the block is, at any position along the shaft, and pick up or drop the block there. This would be difficult to implement because the robot would need to sense where the block is at all times, but it would decrease the limitations of the project significantly.

Source Code

```
#include <Stepper.h>

// define the pins
#define pinEcho 2
#define pinTrig 3
#define startSwitch 4
#define endSwitch 5
#define robotInput 6
#define robotOutput 7
#define LED LED_BUILTIN

const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution for your motor

Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11); // initialize the stepper library on pins 8 through 11

// define the variables
int distance;
int startSwitchHit;
int endSwitchHit;
int potVal;
int finished;


// helper method to calculate ultrasonic distance
int getUltrasonicDistance() {
  digitalWrite(pinTrig, HIGH);
  delayMicroseconds(10);
  digitalWrite(pinTrig, LOW);
  return pulseIn(pinEcho, HIGH) * 0.034 / 2;  // convert pulse duration to distance in cm
}


// helper method to calculate the speed of the stepper based on the potentiometer
// generates values between 200 and 300
int getSpeed() {
  potVal = analogRead(0);
  return (potVal / 10 + 200);
}


void setup() {
```

```
  // initalizing pins
  pinMode(startSwitch, INPUT_PULLUP);
  pinMode(endSwitch, INPUT_PULLUP);
  pinMode(pinEcho, INPUT);
  pinMode(pinTrig, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(robotInput, OUTPUT);
  pinMode(robotOutput, INPUT_PULLUP);

  // initialize the serial port:
  Serial.begin(9600);
}


void loop() {
  // main loop
  distance = getUltrasonicDistance();
  Serial.println(distance);
  Serial.println(getSpeed());
  myStepper.setSpeed(getSpeed());
  startSwitchHit = digitalRead(startSwitch);
  endSwitchHit = digitalRead(endSwitch);
  finished = digitalRead(robotOutput);
  Serial.println(finished);
  if (distance < 3 && finished == LOW) {
    // moves forward with block
    if (endSwitchHit == HIGH) {
      myStepper.step(stepsPerRevolution);
      digitalWrite(LED, HIGH);
      Serial.println("forward");
      digitalWrite(robotInput, LOW);
    }
    else {
      // at the end, calls the robot to pick up the block
      digitalWrite(LED, LOW);
      Serial.println("end switch hit");
      digitalWrite(8, LOW);
      digitalWrite(9, LOW);
      digitalWrite(10, LOW);
      digitalWrite(11, LOW);
      // calls robot to pick up block
      digitalWrite(robotInput, HIGH);
    }
  /}
  else if (distance >= 3 && startSwitchHit == HIGH && finished == LOW) {
```

```
    // moves backward, no block
    myStepper.step(-stepsPerRevolution);
    digitalWrite(LED, LOW);
    Serial.println("backward");
    digitalWrite(robotInput, LOW);
  }
  else {
    // does not move, no block
    Serial.println("running");
    digitalWrite(LED, LOW);
    Serial.println("no movement");
    digitalWrite(8, LOW);
    digitalWrite(9, LOW);
    digitalWrite(10, LOW);
    digitalWrite(11, LOW);
    // calls robot to drop block
    digitalWrite(robotInput, HIGH);
  }
}
```

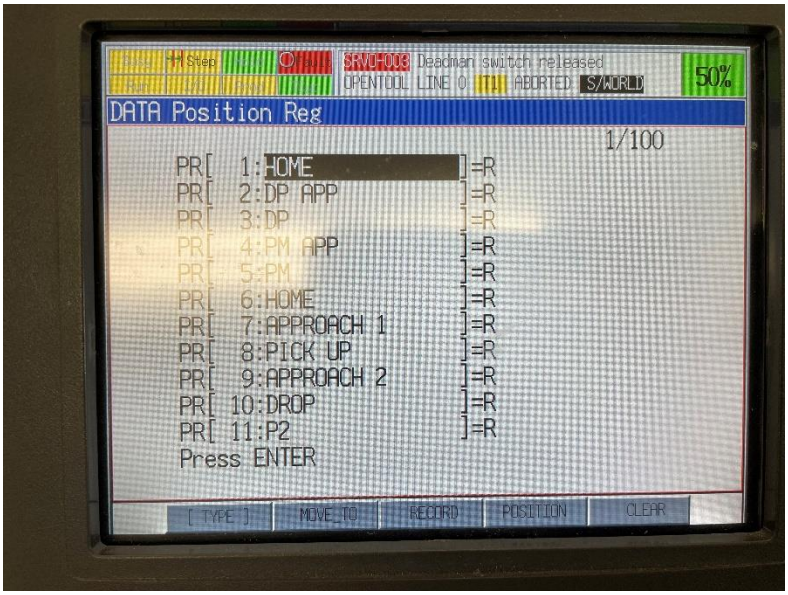

Figure 4: The positions used in the FANUC 200iC Robot programming.

Figure 5: Steps 1 through 11 for the robot function.

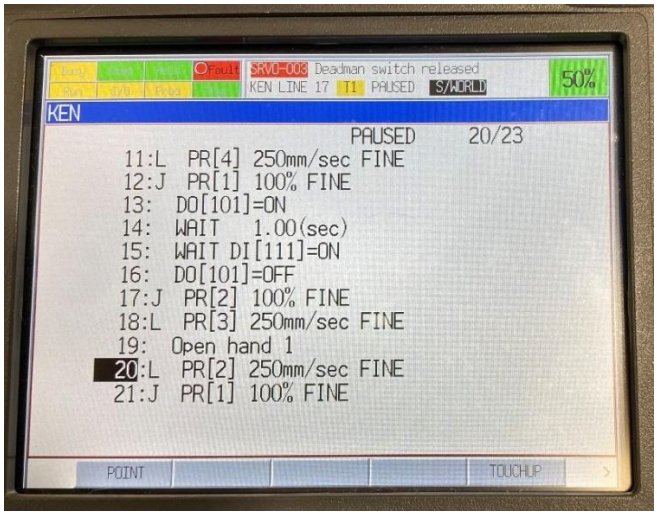

Figure 6: Steps 11 through 21 for the robot function.

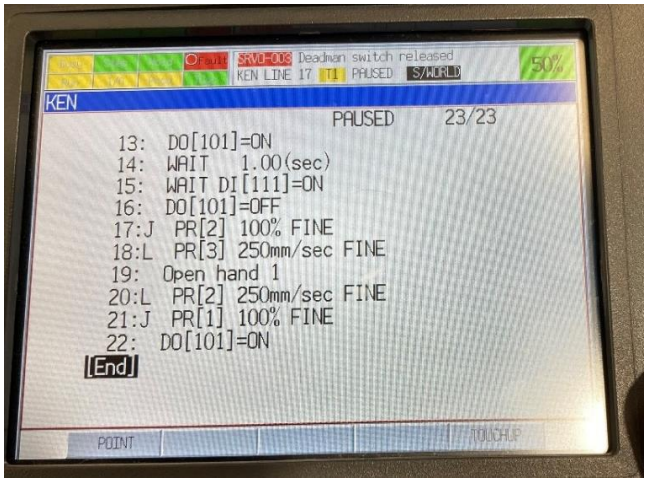

Figure 7: Steps 13 through end for the robot function.

References

[1] *Wokwi - Online ESP32, STM32, Arduino Simulator* (2024, April 27). https://wokwi.com/.

[2] *Miro: The Visual Workspace for Innovation*. (2024, April 27). https://Miro.Com.

[3] *Basics of Potentiometers with Arduino*. Arduino Docs. (n.d.). https://docs.arduino.cc/learn/electronics/potentiometer-basics/.

[4] *Arduino - Limit Switch*. Arduino Get Started. (2024, April 4). https://arduinogetstarted.com/tutorials/arduino-limit-switch.