Process to Achieving a High Fidelity Benchtop Demo for the AFRL Digital Improvement of Propeller Blade Inspection Throughput (DIP IT) Project

Manasa Akella

George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology, Atlanta, Georgia

makella7@gatch.edu

***Abstract*— The objective of this paper is to outline the work done for the AFRL DIP IT project over the Spring 2020 semester. The project goals for this time frame were to create an optimized scheduling algorithm(s) to increase blade throughput for the Warner Robins AFB blade inspection process, create a high fidelity benchtop demo that simulates the process at Warner Robins, and integrate the two to make progress towards implementing the new scheduling algorithms on the FANUC R-2000iC robot at Warner Robins. This paper primarily focuses on the second step of developing a high fidelity benchtop demonstration by learning about and setting up a FANUC robot, first in ROBOGUIDE, and later a physical FANUC LR Mate 200*iD*. The final stage of the project included modeling the setup of the Warner Robins System as best as possible to allow for use of the LR Mate 200*iD* to demonstrate the scheduling algorithm(s).**

I. Introduction

This project, being completed in partnership with AFRL, focuses on the propeller blade testing process at Warner Robins AFB. Every day blades are processed and tested for faults through a process that involves dipping the blades in a fluorescent penetrant dip. The system that the AFB currently uses has been determined to be inefficient. Currently, the AFB uses a FANUC R-2000iC robot that takes the blades through the following process, seen in figure 1 and described below:

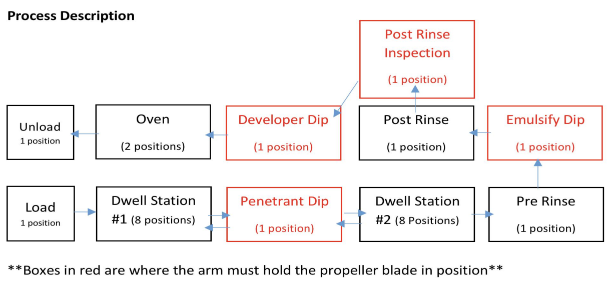

Process Description
Post Rinse Inspection (1 position)
Unload 1 position
Oven (2 positions)
Developer Dip (1 position)
Post Rinse (1 position)
Emulsify Dip (1 position)
Load 1 position
Dwell Station #1 (8 positions)
Penetrant Dip (1 position)
Dwell Station #2 (8 Positions)
Pre Rinse (1 position)
Boxes in red are where the arm must hold the propeller blade in position

Fig. 1 Process Layout

1. Blade manually placed in the Load station (In batches or random singles at random times)
2. Robot takes Blade and places in a Dwell Station (16 slots max)
3. Blade held in Penetrant Dip (PD)
    a. Blade must dwell in Penetrant Dip for at least 4 hours
    b. After PD dwell, must be re-dipped in Penetrant Dip at least once every 60 minutes until "process is complete"
4. Blade placed in Pre Rinse
5. Robot will rinse gripper – (immerse gripper in the Emulsify Dip, then in the Post Rinse)
    a. Rinsed Robot will take Blade from Pre Rinse to be immersed in Emulsify Dip
6. Blade placed in the Post Rinse
7. Blade taken to the Post Rinse Inspection area
8. Visual inspection conducted by operator while dwelling in Post Rinse Inspection - three outcomes:
    a. Blade is "accepted" and moves to Developer Dip
    b. Re-rinse the airfoil (outer part of the blade)
        i. Blade taken to post rinse tank
        ii. Blade returned to visual inspection station
    c. Re-rinse the bore
        i. Blade taken to post rinse tank
        ii. Blade returned to visual inspection station
9. Blade is immersed in Developer Dip
10. Blade is place in the oven
11. Blade is placed in the Unload station

This process occurs with the following times at each station:

1. Average time to dip in emulsify dip
    a. 40 sec dip, 15 sec drip
2. Average time to dip in developer dip
    a. 10 sec dip, 30 sec drip
3. Average amount of time between arrivals of blades to system
    a. Currently this is about 8 mins between blades. The process is currently set up where they load all the blades for the day

(one at a time) and the robot places them in the dwell rack without dipping. Once they are all loaded, the system then dips them all

Using this process and the times listed above, the average blade throughput per day is currently 3-6 blades/day. The goal of the project is to increase the throughput of the process by making it more efficient through the development of reinforcement learning scheduling algorithms. This paper specifically discusses the portion of the project done after creating an initial algorithm, the creation of a benchtop demo to test and demonstrate the capabilities of the scheduling algorithm created. The process to this benchtop demo begins with modeling the robot and cell using the ROBOGUIDE software to visualize the future setup and in order to become familiar with the teach pendant (TP) software to develop a way to run the robot off of the Python code used for the algorithm. The next step includes taking the ROBOGUIDE model and creating a small-scale setup of the Warner Robins process and creating a way to use the smaller FANUC LR Mate 200*i*D to model the process at Warner Robins.

II. Methodology

Creating the benchtop demo had two major tasks that needed to be completed:

1. Learning about and setting up FANUC robot via ROBOGUIDE and
2. Set up of the FANUC LR Mate 200*i*D based off of the ROBOGUIDE setup

The first step was to build a cell similar to that of Warner Robins; However, after running into issues obtaining the exact setup details from Warner Robins a basic setup was created with a simple robot with a standard gripper, a conveyer belt, and a table. The next step was to research and learn how the FANUC teach pendant software worked and use it to code simple motions such as moving from one position to another. Because the operations at Warner Robins had only been run using the teach pendant we also had to determine how to use the teach pendant and integrate it with the new scheduling algorithm written using Python. After much research, a solution using socket messaging and Karel was deemed to be the best solution. The next steps included testing our socket messaging code as well as the reinforcement learning scheduling algorithm with the small scale setup. For this setup, an electromagnet was used instead of a gripper. The biggest task with creating the setup was to design and CAD an attachment for LR Mate 200*i*D to which the electromagnet could be attached which can be seen in figure 2. Using Solidworks I was able to develop an attachment that not only allowed the electromagnet to be attached but also for the orientation to be modified depending on the need. In addition to this, a layout poster of the Warner Robins facility needed to be designed (figure 3). With both of these done, the setup was complete. When the electromagnet was tested it could be seen that it was not receiving enough power when just hooked up to the 5V from the Arduino that was being used to run it. The challenge became developing a way to use the Arduino to wire it so that it could receive power from an external source. In our case, the external power source was a 9V battery. The adapted circuit diagram can be seen in figure 4. After receiving the extra power the magnet was able to pick up the washers that were being used to represent the blades. The next step was to develop a way to turn the electromagnet on and off using the Python code. In other words, the Python code needed to communicate with the Arduino code. Incorporating serial ports was the key to finding a way to make the 'blade' drop or be picked up using user input. After the code was written, the magnet could pick up the 'blade' but would be having trouble dropping it so after testing decreasing the power by 10% increments until it was no longer able to pick up the 'blade' (results of which can be seen in Table 1). After testing that, it was obvious that the reason the magnet was not able to drop the 'blade' was because of the residual magnetism of the electromagnet. By adding a piece of paper under the magnet, as can be seen in figure 5, the blade was able to be picked up and dropped easily using the code seen in figures 6-8

III. Data and Results

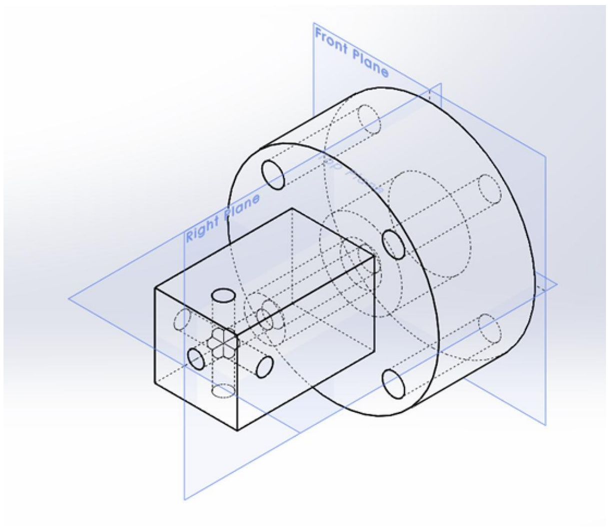


Fig. 2 CAD of Electromagnet Attachment

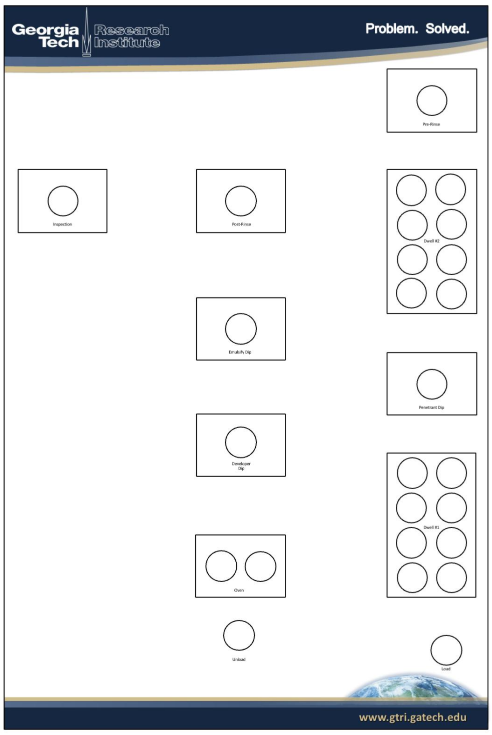


Fig. 3 Poster Layout of Warner Robins

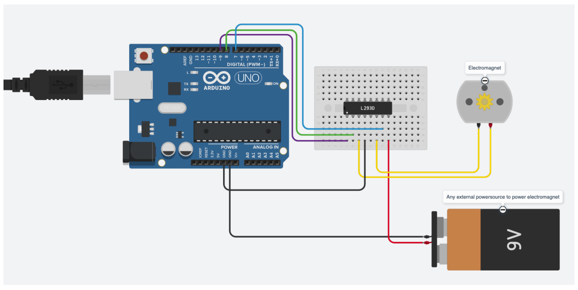


Fig. 4 Adapted Circuit Diagram

TABLE I
Power Level Testing

Power Level	**Results of Drop Instruction at Certain Power Levels**
1	Picks up, drops slightly only when hanging on edge
.9	Picks up, drops when shaken some of the times, drops slightly when hanging on edge
.8	Picks up, drops but still hanging on by the edge (releases the 'blade' but there is still attraction so 'blade' does not fully drop, drops when shaken
.7	Trouble Picking Up 'blade'

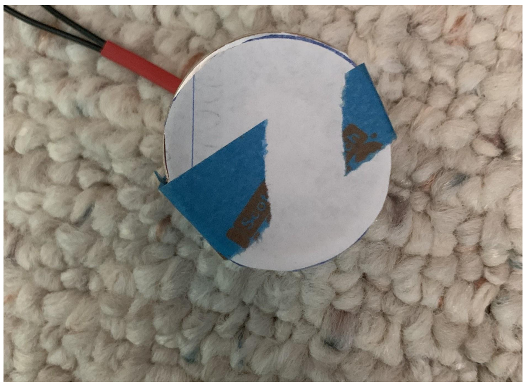


Fig. 5 Paper covered electromagnet

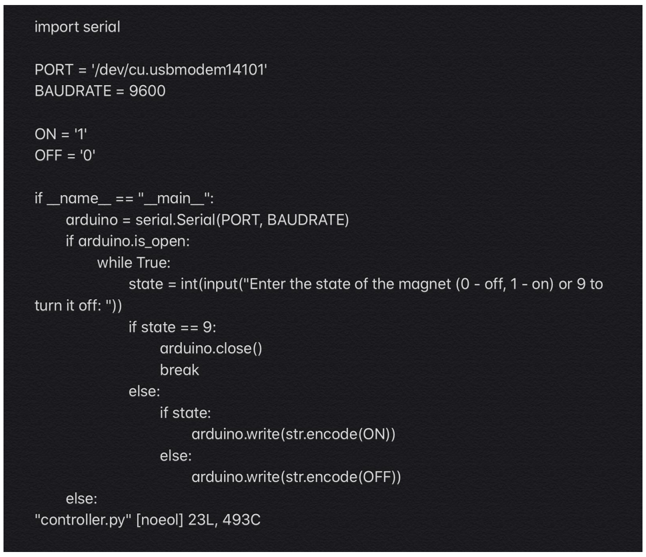
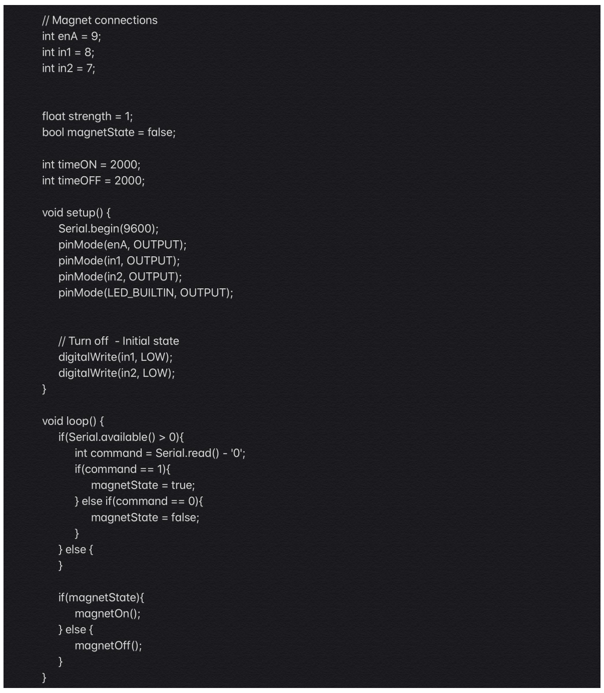
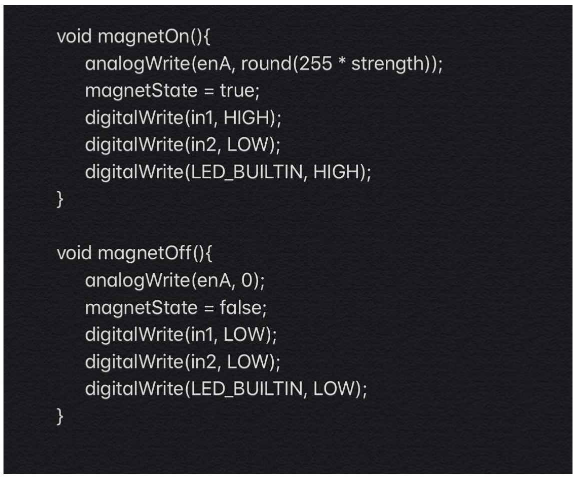

```
import serial

PORT = '/dev/cu.usbmodem14101'
BAUDRATE = 9600

ON = '1'
OFF = '0'

if __name__ == "__main__":
    arduino = serial.Serial(PORT, BAUDRATE)
    if arduino.is_open:
        while True:
            state = int(input("Enter the state of the magnet (0 - off, 1 - on) or 9 to
turn it off: "))
            if state == 9:
                arduino.close()
                break
            else:
                if state:
                    arduino.write(str.encode(ON))
                else:
                    arduino.write(str.encode(OFF))
    else:
"controller.py" [noeol] 23L, 493C
```

Fig. 6 Python Code



```
// Magnet connections
int enA = 9;
int in1 = 8;
int in2 = 7;


float strength = 1;
bool magnetState = false;

int timeON = 2000;
int timeOFF = 2000;

void setup() {
    Serial.begin(9600);
    pinMode(enA, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(LED_BUILTIN, OUTPUT);


    // Turn off  - Initial state
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
}

void loop() {
    if(Serial.available() > 0){
        int command = Serial.read() - '0';
        if(command == 1){
            magnetState = true;
        } else if(command == 0){
            magnetState = false;
        }
    } else {
    }

    if(magnetState){
        magnetOn();
    } else {
        magnetOff();
    }
}
```

Fig. 7 Arduino Code (1)



```
void magnetOn(){
    analogWrite(enA, round(255 * strength));
    magnetState = true;
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(LED_BUILTIN, HIGH);
}

void magnetOff(){
    analogWrite(enA, 0);
    magnetState = false;
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(LED_BUILTIN, LOW);
}
```

Fig. 8 Arduino Code (2)

IV. Conclusions and future steps

Overall, at the end of the project the three overall goals were reached. The robot was able to move using the Python code that was modeled in the benchtop demo using the FANUC LR Mate 200*i*D. The next steps for this project would be to refine the scheduling algorithm based off of the results of testing with the benchtop demo and take further steps to implement it at Warner Robins in the same method it was implemented with the benchtop model