Boston University

EC463 Senior Design Project

Second Prototype Testing Report

NanoPack

A PACKING MACHINE FOR NANOVIEW BIOSCIENCES

BY:

Devin Bidstrup *dbids at bu dot edu*

Joseph Walsh *jwalsh15 at bu dot edu*

Paul Stephen Hutchinson Maltaghati *psmalta at bu dot edu*

Justin Melville *jem2000 bu dot edu*

George Kent-Scheller *georgeks at bu dot edu*

Required Materials:

Hardware:

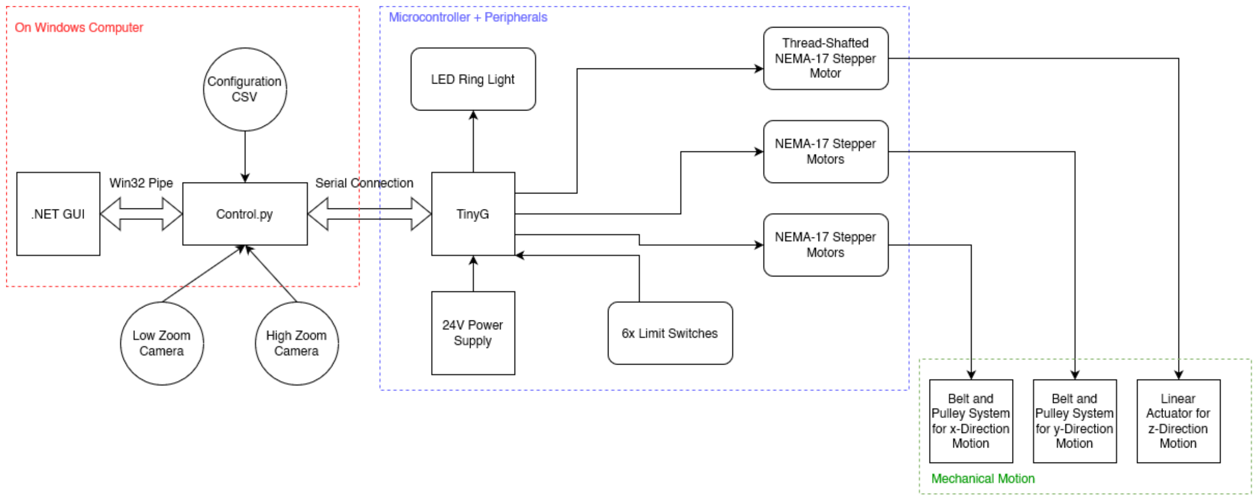
NANOPACK MACHINE

- LitePlacer Robotics kit
    - All mechanical parts
        - Framing, sliders, connectors, etc.
    - TinyG motor controller
    - USB cameras, LED ring lights
    - Stepper motors (3 NEMA 17, 1 NEMA 14)
    - Belts and pulleys
    - Limit switches
    - Cable chain
- 3D printed actuator parts
- Tweezers
- 24V power supply
- Wiring & connectors
- E-stop
- Test Computer

Software:

1. Python control software
2. Machine learning program to process image from wide angle camera
3. Machine learning program to process image from actuator camera
4. .NET GUI for User Interface

Set Up:



The setup consists of both hardware and software components. From the GUI we will run a control program which will move the machine’s actuator from rest to hover over a traveler, whose location will be determined using the large angle camera and a machine learning script. The machine will then use the actuator camera to take a photo and locate chips with the image processing. Then, the actuator will jog down the z-axis, close the tweezers, and return back to the clearance height. The machine will not pick up a chip at this time but will return the location of the chips within the traveler. For this prototype testing, our mechanical goals will be to have the machine moving within the workspace as well as a demonstration of the actuator movement. Our software goals are to locate the traveler and clamshells from the wide angle camera, convert that location to units usable by the TinyG motor control board, send the commands to move the board over it, and to identify and locate the chips within the traveler from the actuator camera.

Measurements Taken:

Criteria	Completed (Y/N)
Image of workspace taken from wide angle camera	Y
Using image from wide angle camera, returns the locations of traveler/clamshell	Y
Machine moves to hover over traveler	Y
Image of chips taken with actuator camera	Y
Using image from actuator camera, returns location of chips in traveler	N
Machine jogs down	Y
Actuator closes tweezers	Y
Machine moves up to clearance height	Y

Results:

1. **Image of workspace taken from wide angle camera**

Even though our CNC machine is not picking up the traveler or the clamshell, we need to be able to detect their locations so that the machine can move to them. To accomplish this goal, we installed a wide angle camera which is able to capture images of the entire workspace. In order to start generating data for our prototype testing, we made a simple camera mount in the corner of the machine workspace. The mount is made out of an 80/20 rod, and it has a webcam bolted at the top of it. The camera, connected by USB to the computer, takes multiple images of the space containing the traveler and clamshells, and then sends these images to the desktop computer. For our testing, the camera was successful at capturing these wide angle images.

2. **Using image from wide angle camera, returns the locations of traveler/clamshell**

After generating the images, we had to write code which was able to determine the location of both the traveler and the clamshells relative to the machine. We wrote a YOLO object detection program in Python which was able to recognize travelers and clamshells. After recognizing the contours of the traveler and clamshells, the YOLO model was able to correctly classify with a confidence of around 0.90. As a result, our machine is able to move to locations within the workspace identified by our YOLO detection. We believe that we can generate higher levels of confidence if we use a better camera which can generate a more accurate flat image.

3. **Machine moves to hover over traveler**

In order to have success at moving to the locations generated from our machine learning code, it was essential that the TinyG microcontroller could be controlled through Python and be correctly homed. To communicate between the TinyG and the control script, we created a serial connection Python script which allowed our lab computer to connect to the TinyG, which runs off of the G-code. After establishing connection, the Python script then began a configuration and started homing. Through the use of wired limit switches, homing allowed for the CNC to establish an origin at the corner of the workspace, and from this origin it was able to move to a location determined by the machine learning algorithm. Being able to correctly configure and home the machine required a lot of effort due to the fact that the TinyG has dozens of parameters which have to be set. The parameter setup was done using JSON files, and the information in these files was sent through the Python serial communication to the TinyG. Moving the machine successfully also took a lot of work mechanically. We received our CNC kit from our client at the beginning of the semester, and we were able to build and wire it quickly to ensure that we had enough time to learn and work with the motor controlling G-code. A minor mechanical issue which we will be addressing is the impact which homing has on our mechanical belts. When hitting the limit switches, lots of force gets placed on the belts, and with continued use they may become looser. We are now working to fix this issue for the final product. Overall, though, we were successful in completing this measurement.

4. **Image taken with actuator camera**

In addition to the wide angle camera, our machine also has a camera on the linear actuator which is responsible for taking close-up images of chips within the traveler. This is done in our control script by using the OpenCV video capture method. This image is then saved locally.

5. **Using image from the actuator camera, returns location of chips in traveler**

While we were able to return the location of travelers and clamshells for this prototype testing, we were unable to locate the positions of individual chips within the traveler. At the time of the testing, we were still refining our chip recognition code. The chips were difficult to recognize due to the edges within the traveler and the reflective nature of the workspace surface. After doing some additional work, we are now able to recognize and extract individual images of each chip from the traveler, as well as digits within the chip number label. There is still the issue of the lighting, which is something preventing very precise locations from being determined. Another thing which was brought up when preparing for our prototype testing is the fact that our chip identification script only processed images which weren’t angled. For our testing, we restricted the traveler location to resolve this.

6. **Machine jogs down**

For the machine to jog down, it requires our z-direction motor to work according to the direction it gets from the TinyG. The TinyG received its instructions from a function defined within our control script as well as the Python TinyG serial connection code. The TinyG was then able to successfully communicate with the z-motor and get it to jog down the correct distance.

7. **Actuator closes tweezers**

To pick up our chips, our team designed our own actuator which uses a smaller motor to close a set of tweezers which pinch and hold the chips as they are transported. The motor came in the kit supplied by our client. The actuator was designed in CAD and 3D-printed. The size of the actuator had to be adjusted a few times, but by the time of the second prototype testing, we were prepared to close the tweezers by using the actuator. The tweezers were attached to the actuator through zip ties. The TinyG commands to close the tweezers were also called in our control script. While we did not pick up a chip during the testing, we were able to demonstrate that with a bit of tuning, we would be able to.

8. **Machine moves to clearance height**

After picking up the chips, the z-direction motor was required to make upward movement occur. As for the other motor control actions, the movement was the result of our control code calling the TinyG through the serial connection. This final part of our prototype testing was a success.

<u>**What’s Next**</u>

<u>Graphical User Interface</u>

Most of the work for our GUI has been completed, as we can upload CSV data files to it and begin the packing process through the click of a button. The remaining work which we have for the GUI is ensuring there are no issues with the pipe. While our system has become a lot more robust, we will need to make sure that pipe does not break when our entire project is done. This will require lots of repeated trials and testing.

<u>Machine Learning Models</u>

At this point, we have now developed machine learning algorithms which can detect and classify travelers and clamshells from images as well as chips and chip label numbers. These are some things which we will be doing to make improvements:

a. **Retraining our algorithms with data from new hardware**
   Even before our prototype testing, we knew that we wanted to purchase a better camera and host it in a more ideal location. Right before our spring break, our team purchased a camera which does not require images to be flattened. Now, we will have to create a new mounting system. We plan to place this mounting system at the side of the center of the workspace. After installing this new camera at its new location, we will have to generate new image data and retrain our traveler and clamshell location model.
b. **Identifying traveler chip locations which can be sent to the TinyG**
   It is very important that our chip recognition software is able to determine the locations of individual chips and send these locations to the TinyG. We will have to make sure that these locations are very precise since the chips are packaged very close to each other on the traveler.
c. **Determining clamshell locations and relating these locations to chips**
   We will have to train our models to recognize locations in the clamshells and determine which chip corresponds to which location. Since there are more contours within the clamshell, our algorithms will have to be adjusted.
d. **Recreating numbers from individual digits**
   Since we have now been able to extract individual digits, we will now have to recreate the numbers from these digits. These digits will have to be reconstructed if we want to be able to compare these numbers to the numbers within the uploaded CSV file and extract the related clamshell locations.
e. **Doing lots of testing**
   Once everything mechanically is attached to the machine and all of the hardware improvements have been made, it will likely require a lot of testing to ensure that the final product works, especially with regards to picking up and placing individual chips.

TinyG Motor Controller

In regards to how we control the motors with the TinyG board, the main thing that we need to do is to create reading and writing threads which synchronize to access the serial connection. This will allow us to always have a reading thread running to clear the input buffer for the serial connection on the computer, ensuring that we receive all messages even if they take some time to generate or occur during moves. This also will allow us to detect when the machine is done with a given command so that we can issue a new command with little latency, greatly reducing the time it takes to do tasks. As of the time of writing this report, this is close to completion, though it was not implemented for the second prototype test.

Mechanical

For the mechanical design, the main area of work will be in installing and testing our actuator and making sure that enough torque can be applied to pick up our chips. We will be changing our actuator motor from a NEMA-14 to a NEMA-17 to ensure that torque is not an issue. We also will be painting our workspace to reduce the reflectiveness of its surface. We will also be adding restraints to hold the clamshells/travelers at a right angle so we will be able to consistently pick up the chips, as the actuator does not rotate.