**Boston University**

**Electrical & Computer Engineering**

**EC464 Capstone Senior Design Project**

**User's Manual**

Submitted to

NanoView Biosciences

1380 Soldiers Field Rd Ste 1000,

Brighton, MA 02135

by

Team # 33

NanoPack

Team Members

Devin Bidstrup dbids at bu dot edu

George Kent-Scheller georgetks@gmail.com

Justin Melville jem2000@bu.edu

Joe Walsh jwalsh15@bu.edu

Paul Stephen Hutchinson Maltaghati psmalta@bu.edu

:

Submitted: 4/15/22

#### Table of Contents

[Table of Contents](#_heading=h.y6ofvpgn7vw) 1

[**Executive Summary (Author: Joe Walsh)**](#_heading=h.4lmlgji7n6yr) **3**

[**Introduction (Author: George Kent-Scheller)**](#_heading=h.1fob9te) **1**

[**System Overview and Installation**](#_heading=h.3znysh7) **3**

[Overview block diagram (Author: George Kent-Scheller)](#_heading=h.2et92p0) 3

[User interface. (Author: Justin Melville)](#_heading=h.tyjcwt) 3

[Main Page:](#_heading=h.o92ds7fudii8) 3

[Traveler View](#_heading=h.y4uedwnoaa50) 3

[Camera View Page](#_heading=h.q4vqab1938yl) 3

[Console Debugging Page](#_heading=h.w4dsf6jhebrh) 4

[Manual Port Selection](#_heading=h.51n8qowdplph) 4

[Physical description. (Author: George Kent-Scheller)](#_heading=h.3dy6vkm) 4

[Installation, setup, and support (Author: Devin Bidstrup)](#_heading=h.1t3h5sf) 4

[Notes](#_heading=h.h7kgdmjh69y0) 4

[Installing NanoPack App](#_heading=h.rp0wg8jqgaa7) 5

[Clone the GitHub repository](#_heading=h.zgvbjjwbpko5) 5

[Python Environment Installation](#_heading=h.vq0x8gqb5uwy) 5

[Run the Application](#_heading=h.kbd0mex9qc17) 5

[Physical Installation](#_heading=h.kbmumrifr9m2) 5

[Camera Tests and Configuration](#_heading=h.xohv6y9pcw28) 5

[Wiring Setup](#_heading=h.chh2v37mhk80) 6

[Limit Switch Alignment](#_heading=h.ogds2zej7ia) 6

[**Operation of the Project (Author: Justin Melville)**](#_heading=h.4d34og8) **7**

[Operating Mode 1: Normal Operation](#_heading=h.2s8eyo1) 7

[Operating Mode 2: Abnormal Operations](#_heading=h.17dp8vu) 7

[Safety Issues](#_heading=h.3rdcrjn) 7

[**Technical Background (Author: George Kent-Scheller)**](#_heading=h.26in1rg) **8**

[**CNC Control**](#_heading=h.zia5o0xfllnw) **8**

[**YOLO ML**](#_heading=h.kmu1abpp6f3j) **8**

[**Number Recognition ML**](#_heading=h.qnpoj25vj861) **8**

[**Image Processing**](#_heading=h.lpq6wi4x3id4) **9**

[**Actuator Design**](#_heading=h.mderbrl32ndc) **9**

[**Actuator Control**](#_heading=h.6dg2nbonyly1) **9**

[**Custom Wiring**](#_heading=h.mzn1posuq9lb) **9**

[**Relevant Engineering Standards (Author: Paul Stephen Hutchinson Maltaghati)**](#_heading=h.lnxbz9) **10**

[Coding Standards](#_heading=h.d98hpncx3axk) 10

[Electrical Standards](#_heading=h.c1olo7vpulxx) 10

[Robotics Standards](#_heading=h.u81e1ckbceb7) 11

[**Cost Breakdown (Author: Joe Walsh)**](#_heading=h.35nkun2) **12**

[**Appendices (Author: Devin Bidstrup)**](#_heading=h.1ksv4uv) **18**

[Appendix A - Specifications](#_heading=h.44sinio) 18

[Appendix B – Team Information](#_heading=h.2jxsxqh) 18

# 

# Executive Summary (Author: Joe Walsh)

NanoView Biosciences is a startup company dedicated to making research equipment to further explore exosomes, extracellular vesicles, and viruses. As the company expands and offers its services to more customers, it will need to ramp up the production of its signature product, a specially treated silicon chip that allows its customers to analyze biological samples. The current production process for these chips includes a step where a lab technician must use tweezers to manually move chips from an internal carrier of 64 chips (10 x 10 x 0.5 mm) called a traveler into packages of four for shipping called clamshells. This is both labor-intensive and error-prone. The NanoPack machine is our proposed solution to this problem. The machine is modified from an existing “pick and place” (PnP) machine used to manufacture PCB boards. The actuator was custom-designed to operate a pair of tweezers (just as the technician does) to pick up and place the chips in their appropriate slot in the packaging. The machine is given an input .CSV file with chip information. It then utilizes computer vision and machine learning to locate the traveler and then to move over it where a custom CNN does digit recognition for the chips. Once the machine has determined that the correct chips are in the traveler it will place the scanned chip into its corresponding clamshell which is located with the same machine learning algorithm that found the traveler. While all of the chips in the clamshells are the same, there are different kinds of chips in each traveler. We estimate that this will be able to increase the company's throughput of chips by ten times. Additionally, all of the software to support our modifications and the machine learning models will be packaged with a GUI and turned into a ready-to-go Windows .exe. This will allow NanoView to use the machine easily and without a large amount of configuration between runs. The remainder of this manual will describe how to set up and use NanoPack as well as our technical approach to building it.

# Introduction (Author: George Kent-Scheller)

1. ***Problem***

In the field of bioscience, there is a demand for precise portable cell imaging technologies. Our client, **NanoView BioSciencesTM**, has developed a proprietary imaging technology that allows labs to either take in-house samples and then process them with the help of a specialized machine, or send them to *NanoView* for analysis. This imaging process involves placing the desired biological medium onto a specialized silicon chip and then inserting the chip into the machine for analysis. The machine is capable of detecting exosomes as small as 50 nanometers and can identify up to 5 separate biomarkers on the sample using fluorescent light and dye. This technology allows *NanoView* to command a competitive advantage in their field. Hence, the company must expand production in the future.

Currently, *NanoView* is experiencing a manufacturing bottleneck. To get the product to the market more efficiently, *NanoView* started off doing most of their production by hand. As they have grown, they have attempted to increase throughput at each step in their production line. To decrease bottlenecks, they have decided to have a central tray that carries chips between manufacturing processes, called the traveler. The manufacturing processes are conducive to this model as many of them involve scanning the chips or coating the chips (i.e. the chips do not need to be removed from the tray). One of the final processes in the manufacturing of the silicon chips is packaging these chips to send out to the customer. This step is causing a restriction on the output which our project aims to alleviate. Currently, chips are pulled from the tray by hand, checked visually against a spreadsheet, and then placed in their corresponding packaging. This process is vulnerable to human error, ties up employees on menial tasks, may damage the chips, and most importantly is slow.

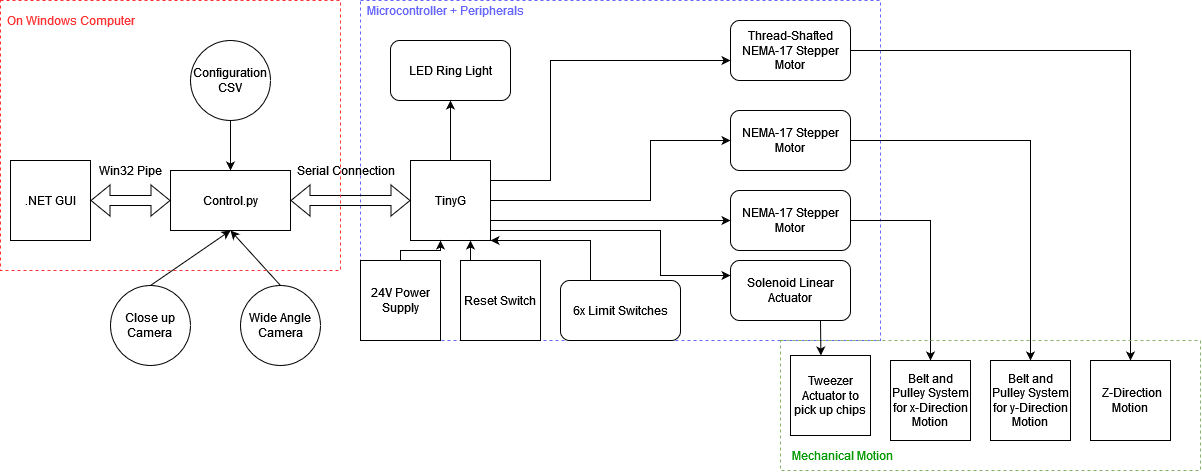
1. ***Our Solution***

We aim to automate the final packing step, thereby increasing the speed at which it is performed. To automate this process we will provide *NanoView* with two things: a packaging machine and a software program. The packaging machine will be a CNC-style device capable of moving along three axes. Fitted to the end of the moving arm will be a tweezer-like actuator which will be used to transport the chips, making sure to grab the side of the chip to avoid damage. Attached to the machine will be two cameras, one to capture a broad view of the entire device and one attached to the arm to capture the chip number at a given point in time. This will allow us to leverage machine vision to capture in real-time the chip numbers that the arm is moving, and the positions of the chips, traveler, clamshell, and arm. The software program will provide the user with the ability to start the packing, monitor its status in real-time, detect errors, and terminate it if need be. The software package will be portable so that it can be installed on any Windows device. With our system, the lab technicians will be able to insert a full traveler and empty open clamshells, open the software package, load a CSV with the chip information, and begin packing. This will allow *NanoView* to pack with far greater efficiency than before. In addition, should the machine detect discrepancies between the chip numbers and the CSV, it will alert the user. In this way, we provide *NanoView* automated packing and error detection for a part of their manufacturing process.

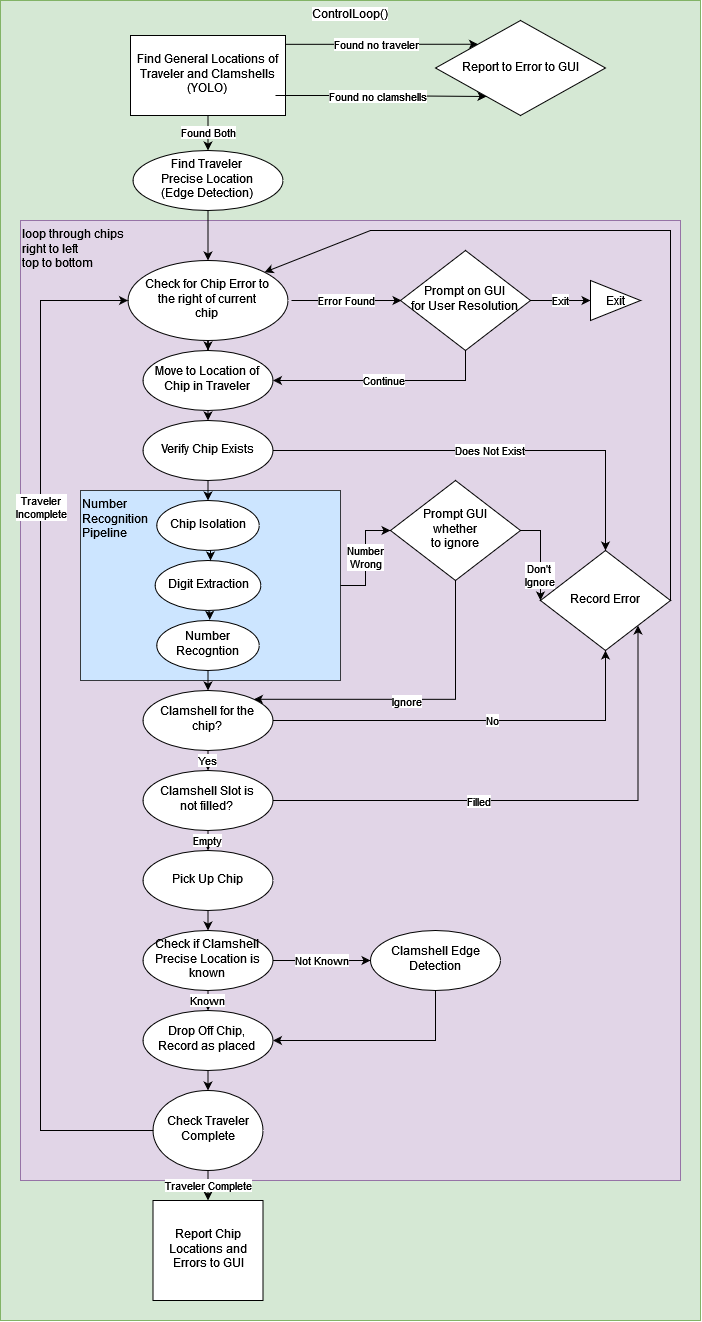
# System Overview and Installation

## Overview block diagram (Author: Devin Bidstrup)

Our system uses a .NET frontend that communicates with a Python backend. Within the python backend, ML models perform number recognition and object detection, commands are sent to a GCode-based motor control board, and the main control script is executed. The Python process communicates with the .NET front end with a pipe and talks to the TinyG over a Universal Serial Bus.



The Python control module runs through a state machine to execute the packing of chips.



*Note: there is a physical constraint in the traveler which necessitates that we pick up chips from right to left for a given row. Ideally, we would have designed our loop to skip over chips that error, and alert the user about them after packing, but if a chip errors to the right of another chip, we cannot pick up that chip on the left due to this constraint. As such, the user will be prompted in these moments to decide whether to kill the packing process or ignore the chip that it cannot pick up.*

## User interface. (Author: Justin Melville)

For the user interface, we built a Windows Form application in C# .NET. This user interface is capable of being run on Windows, matching the specification set for us by our client, Nanoview Biosciences. The UI starts the python backend, and thus is the only script that needs to be run to use the NanoPack machine. Once opened, the UI has a total of three pages.

### Setup Page:

On the setup page, there is only one task to complete. This task is selecting the path to where the activate.bat script is for your installation of Anaconda. Once this is provided, you can press continue to go to the main page of the application. Once this process has been completed, it should not need to be repeated on subsequent runs of the program.

### Main Page:

On the main page, there are three buttons.

1. Upload file - Uploading the file must be done first. Simply click the button, which will open your system file explorer. The chosen file must match the format set by NanoView. Once the file is uploaded, the data will be visible in a table, and a traveler visualization will be loaded.
2. Initiate Packing - This button will send the chosen .CSV to the backend and start the packing process.
3. Cancel - This button cancels the packing process by sending a stop command to the backend. Note: This button is only designed to work if the machine is operating normally. In case of error, the physical emergency stop button should be pressed instead. This can be found on the back of the microcontroller container.

### Traveler View

Upon .CSV upload, a visualization of the traveler will be loaded. It is useful for ensuring the traveler and the .CSV match as you can easily compare the position of the chips within the grid displayed on the UI and those within the physical traveler.

### Camera View Page

On this page, you will see a button called "Load Images from Machine." Pressing this button will enable you to view the pictures of the chips taken by the camera. These pictures will continuously update as the machine runs. In order to disable this stream, simply press the button again.

### Console Debugging Page

Again, this page is not necessary to complete the packing process. It allows you to see the messages being sent by the backend to the frontend. This is useful if you want to see packing status reports. It is also useful for seeing how many iterations are completed before an error occurs.

### Manual Port Selection

Manual port selection is something that should never need to be dealt with, as NanoPack automatically attempts to detect the Windows COM port it is plugged into. However, in the event that automatic detection fails, the UI will display a popup message manually allowing you to select which port to use.

## Physical description. (Author: George Kent-Scheller)

Our client will receive a system consisting of two principal components. Firstly we will deliver a physical machine that will be capable of packaging a silicon wafer chip into a set of containers. This machine will be constructed primarily out of aluminum extrusion bars which are fastened together. It will be able to fit in our clients facility by having dimensions that do not exceed 2’ x 2’ x 2’. There will be an interior area called the packing bed that will be a flat surface with dimensions of 2’ x 1.5’. The machine will be able to move an actuator with 4 degrees of freedom within the packing bed. This movement will be powered by a series of NEMA stepper motors all running on 12 v DC. They will be controlled by a series of motor drivers taking input from a microcontroller with a large number of output pins capable of creating a 5 volt PWM signal. These motor drivers will be powered by a 50 watt 12 volt dc power supply running off of 120 volts 60 hz ac. The microcontroller will also have an lcd panel to display the current progress of the packing procedure. There will also be two emergency stop switches. A first which sends a software interrupt to the arduino to allow it to fail into a safe state, and another which cuts all the power to the motors. The movement of the actuator will be controlled in the Z axis by a servo connected to the microcontroller to control Z axis rotation as well as a microstepper linear actuator to control Z axis elevation. The actuator will grab the chips, with dimensions of 9.5mm x 9.5mm x 2.5mm, by closing a moving arm onto a fixed arm to create gripping force through the use of either a stepper motor or a servo. This force shall not exceed 520 N to avoid deforming the silicon wafer.

## Installation, setup, and support (Author: Devin Bidstrup)

## Notes

* *This application will only work on Windows devices*

## Installing NanoPack App

### *Clone the GitHub repository*

If you already have git installed ignore this section.

* Download [Github Desktop](https://desktop.github.com/).
* Follow the tutorial [here](https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository) released by git to guide you through cloning a repository with the URL of this repository instead.

### *Python Environment Installation*

* Install Miniconda using the installer found [here](https://docs.conda.io/en/latest/miniconda.html).
* Once Miniconda is installed, search on the taskbar for **Anaconda Powershell Prompt** and open it.
* Navigate to where the GitHub repository has been cloned using: cd <REPO\_PATH>
* Then run the following commands:
  + cd .\build\
  + conda env create -f condaconfig.yml
* Finally test the installation by running the command conda activate NanoPackEnv.



### Install *the .NET Application*

* Our application is built with .NET to the following folder: <REPO\_PATH>\dev\NanoPackUI\'NanoPack UI (draft)'\bin\publish
* In this folder, you should run the **setup.exe** file to install the app on your device, so that you can run it like any other app; from the search icon.
* If this application runs successfully, then you have installed the NanoPack App.

## Physical Installation

You should be provided with the physical machine, though on arrival it will require some tuning and validation to ensure that everything is working correctly.

### *Camera Tests and Configuration*

The overhead camera can rotate somewhat, which could seriously impact the ability of the machine learning algorithms to find the traveler and clamshells. First, ensure that the USB cables for the two cameras are connected to the Windows PC and open the Camera app. You should see both a close-up angle from the perspective of the gantry and (if you switch the camera views) a wide-angle view of the whole setup. The bed of the machine is green to support the machine learning so verify that only the green portion of the bed is captured. Please ensure that the overhead camera is angled to capture only the usable space.

### Wiring Setup

The TinyG board should already be wired to the machine and placed inside its enclosure. Take a minute to ensure that three stepper motors are connected to the board, with their 4 pin connections placed correctly. A relay should be within the enclosure and connected to the SprDir pin, 3.3V pin, GND pin, and 12V pin to drive the actuator. Additionally, ensure that the power supply is connected to the board. There should be a standard US power cable with three wires connected to the power supply, which will be plugged into the wall to turn on the machine before running any programs. Finally, there are six limit switches, four on the gantry which should be wired through the cable chains to the TinyG control board, and two on the outside of the machine also wired to the TinyG control board.

### *Limit Switch Alignment*

The last physical devices to configure are the limit switches. The limit switches are designed to hit the machine and trigger just before the machine would collide with the housing. It may be necessary to loosen the bolts, adjust the limit switches, and then tighten them again. However, doing so will change the machine's zero-point thereby introducing an error. This error could be avoided by re-tuning the code, but this is a lengthy and complex process. Making these limit switches as precise as possible will greatly help the program function consistently.

# Operation of the Project (Author: Justin Melville)

### Set Up

Prior to operating NanoPack for the first time, set up must be completed. To begin, simply run NanoPack for the first time. This will show a page that asks for the path to the activate.bat script for your installation of Anaconda. Once this path has been provided, pressing continue will finish the process and load the main page of the program. Once this process has been completed, you will not need to repeat it unless you use a different computer.

### Operating Mode 1: Normal Operation

Once NanoPack is fully set up and operational, getting started with the packing process is quite simple. Please ensure the lights in the room are on to ensure NanoPack can take pictures of the chips. On the bed of the machine, you will see a space labeled "traveler" and a space labeled "clamshells." Please place the traveler and a sufficient number of clamshells in their respective sections in NanoPack. Both the traveler and clamshell should be oriented in the same direction (facing the user). Once this is done, you are ready to begin packing.

In order to start the packing process, simply run the NanoPack.exe file. From there, you will see a user interface with some different options. The only necessary steps to initiate packing are to:

1. Upload CSV
2. Initiate packing

Once these steps have been completed, NanoPack will begin packing chips. Lastly, you will receive a message on the user interface stating that the program exited with the exit code, "NormalExit."

### Operating Mode 2: Abnormal Operations

In the event that NanoPack cannot find the specified chip in the location specified by the chip location .csv file, it will scan the entire traveler to look for it. If it cannot find the chip anywhere on the traveler, it will simply skip this specific chip and continue on with the operation of the machine. The user will be notified at the end of any missing chips. Any chips in the traveler that are not in the .csv will simply be left in the traveler. We recommend that if any leftover chips are still in the traveler after operation, that they be moved to their proper clamshell by hand. In the event that this occurs, you will receive a different exit codes in the user interface other than "NormalExit." These exit codes specify why the machine did not run properly, e.g., "TooFewClamshells," "TravelerNotFound."

## Safety Issues

Although NanoPack does not move with enough force to cause serious bodily injury, we still recommend that you do not place any body parts in or above the operating bed once packing has begun. It is only safe to retrieve the empty traveler and filled clamshells once you have received a message on the UI stating that packing is complete.

# Technical Background (Author: George Kent-Scheller)

Our machine packs chips into clamshells by using a heavily modified, circuit-board - placing, C.N.C. (computer numerical control) machine in tandem with AI. It also uses computer vision to increase accuracy.

### CNC Control

The C.N.C machine that we used for our project is intended to be used to place circuit components on circuit boards before they are soldered. It came in a kit, but only included the mechanical components. Our team did all of the complex wiring and assembly ourselves. The machine also comes with accompanying CNC software. This is software that lets a user input a particular pattern of circuit components via a Graphical User Interface and turn that into instructions for where to place chips. The software that comes with the kit automatically turns the instructions from the GUI into a machining language called G-code. These instructions is then sent from the computer to the controller CNC module via a USB cable. G-Code is a low level language that is hard for humans to conceptualize but is an industry standard for CNC machines. Typical CNC machines are rigid, only cutting a predetermined path. However, our modified CNC machine adapts to the environment and can move to pick up the chips in many different locations. We wrote software that takes our movement algorithms and turns them into G-code commands before sending them over USB to fix this problem. This software replaces the software included in the kit which did not suit our purposes.

### YOLO ML

We also are able to detect many things using AI and Computer Vision. Using a camera we take an image of the whole bed. Then an AI algorithm ([yolov5](https://github.com/ultralytics/yolov5)) that is used for finding objects within a picture is run to find the location of the traveler. This algorithm is a pre-existing machine learning model that we re-trained using our data. However, due to changes in light, movement of the camera, slippage of the belt, and general bad luck, the position that the machine learning gives is not accurate enough. However, this location does give a good point from which to start. When the location is found, we move the grabber head over the traveler. This head is equipped with another camera and takes pictures of what is below it. Using photo-processing techniques, it is able to tell where the edges of a certain object are and then find the top of the traveler. We are then able to repeat this step as we pack for each of the clamshells.

### Number Recognition ML

Then we use AI again to read the chip numbers off the top of the chips. This is a hard problem because the chips have a reflective coating and numbers which are not standard. We used a Convolutional Neural Network with four convolutional layers and one max pooling layer. This type of image classification AI is useful for classifying digits into different categories. We created this model. Both models run in a lightweight deployment environment with pre-trained weights making them faster than easier to implement machine learning algorithms.

### Image Processing

For both of these computer vision tasks we do extensive image processing. (LINK TO GITHUB PYTHON FOLDER AFTER CONSOLIDATION OF CODE BASE)

### Actuator Design

Mechanically, we have designed a custom actuator for the head of the machine. This actuator is used to pick up the chips after the software has located them. It is 3D printed and holds the solenoid that was custom wired for the machine. The parts geometry is designed to allow the machine to move further to the left by adding a cut-out for the belt pulleys. Additionally, it can hold the tweezers steady with a self-ratcheting 3D printed shim. There is also a custom camera mount to hold the wide angle camera used for machine learning. There are also custom built and mounted restraints to keep the lids of the clamshells open and to keep both the traveler and clam shells perpendicular to the actuator.

### Actuator Control

Apart from motors included in the kit, we also use a solenoid valve to close the tweezers. This is a good solution because the solenoid has enough force to hold the chips in place but would hurt the chips directly. The flexibility of the tweezers allows them to firmly grip the chips while deforming themselves to lessen the force experienced by the chip. The solenoid is controlled by a 12 volt fan pin coming from the CNC controller that is toggled on and off using a relay that is controlled by 3.3v logic.

### Custom Wiring

Additionally we included our own wiring for all the controls. All of the wiring is shielded so that the inductance generated from one stepper motor does not interfere with the accuracy of the others. Each of these motors also has easy disconnect switches so that it is relatively similar to replacing a burnt out motor, or a single bad wire.

# Relevant Engineering Standards (Author: Paul Stephen Hutchinson Maltaghati)

Since NanoPack is a CNC machine requiring the usage of hardware and software, our team followed standards for coding, electrical systems, and robotics. We followed these standards in order to ensure that our product documentation is easy to follow and understand. We also wanted to make sure that our machine operates safely within its operating environment.

### Coding Standards

We wanted to make sure that we followed IEEE coding standards. To accomplish this, we made sure to split our GitHub repository into different folders based off of development code, source code for official testing, and non-code development materials such as images and and schematics. Within our development section, we separated each workflow into their own folders, with the main ones for our project being machine learning, the TinyG embedded system, and the user interface. Our group also followed these standards when writing individual scripts. For example, all of our Python code follows standard with regard to the naming of functions and variables.

### Electrical Standards

For the electrical standards, our team followed the IEEE National Electrical Safety Code (NESC). NanoPack requires a power supply, a microcontroller, motors, and limit switches to work electrically, and all of these components had to be connected via wires in a safe and efficient way. By following NESC, we ensured that NanoPack is safe for its users. Our project involved a lot of wiring, as we had to connect all of our motors and wires to the TinyG microcontroller. Each of these components required multiple wires, and each of them were located at different places with respect to the TinyG. All component connections also required different lengths of wire connectors. In order to make the wiring more efficient and safe, we shielded all of the wires. We also added connectors to all wires to make disconnecting easier and safer. If any components need to be replaced or modified, people working on the project can simply separate the wires and make necessary changes. Another electrical standard we followed involved the safety of our microcontroller. When fully powered, the TinyG is able to supply 12V. If a person were to touch the open connection, they would experience a powerful shock which could potentially be harmful. To avoid any potential injury, we enclosed the TinyG. This enclosure can also be easily opened when changes to the board need to be made. Another electrical standard which we set for our project was the use of an emergency stop button connected directly to the microcontroller. In the event that an emergency occurs where the machine must be immediately cut from power, there is a button on the side of the enclosure which, when pressed, interrupts the connection from the power supply to the TinyG. When this is pressed, the machine stops everything that it is doing, and no current flows anywhere in the system. A final standard which we are incorporating into the design is the grounding of the workstation. With the continuous operation of the machine, static charge is built up on the metal of the workstation. While this charge is extremely unlikely to have any noticeable effect on a person, it could impact electrical connections. It is best practice to ground any workstation, which is why we incorporated.

### Robotics Standards

We put safety measures in place for our robot itself, as we wanted to ensure that no mechanical aspect of our machine had the potential to cause any harm. By following standards of IEEE robotics and automation, we ensured that the CNC is in an enclosed space. By using limit switches, we ensured that the machine will stop once it reaches the edges of the workspace. In the construction of the CNC, we followed the assembly instructions provided to make sure that all components were secure.

# Cost Breakdown (Author: Joe Walsh)

When creating the *alpha* version of our machine, we were provided with a LitePlacer CNC kit to modify. This kit costs $1799 retail and contains most of our actual materials. The prices we found for the items in the table below come from a variety of sources based on the part numbers and manufacturers listed in the LitePlacer's Bill of Materials. The dominating costs from outside the LitePlacer kit include the power supply, the machine bed, and the wide-angle camera being used to take an overhead shot of the workspace. Some materials were also sourced from the Senior Design lab (e.g. wires) and spare parts (solenoid). We also could not find prices for some of the provided aluminum parts, so we decided to use their identical geometries and get price quotes for them to be 3D printed (in PETG). The custom-designed actuator parts as well as any other 3D printed parts had their price calculated based on weight. The total price of the *beta* version of our machine is $1259.32.

| **Item** | **Quantity** | **Description** | **Unit Cost** | **Extended Cost** |
| --- | --- | --- | --- | --- |
| **Electronics** |  |  |  |  |
| TinyG v8 | 1 | Computer | $165.00 | $165.00 |
| 350W 24V 14.6A 115/230V Switching Power Supply Stepper Motor CNC | 1 | Power Supply | $29.18 | $29.18 |
| Omron D3V-162-1C4 | 6 | Limit switches | $3.19 | $19.14 |
| NEMA17 stepper motor | 2 | Motors | $10.13 | $20.26 |
| Nema17 stepper motor 40mm body 150mm threaded shaft | 1 | Motors | $28.99 | $28.99 |
| TAU-1040t solenoid | 1 | Actuator | $4.95 | $4.95 |
| Huiber HBV-1517 | 1 | Camera | $6.00 | $6.00 |
| Wide-angle camera | 1 | Camera | $33.99 | $33.99 |
| Connectors | 24 | Wiring | $0.01 | $0.24 |
| Cable carrier | 2 | Wiring | $10.20 | $20.40 |
| 22 gauge wire | 40ft | Wiring | $0.09 | $3.60 |
|  |  |  | **Section Total** | **$331.75** |
| **Framing** |  |  |  |  |
| Misumi HBLFSNK5 single side tab bracket with side supports | 4 | Brackets | $1.05 | $4.20 |
| Misumi HBLSS5 thin bracket | 7 | Brackets | $1.05 | $7.35 |
| Misumi HBLTSW5 bracket thick, 90deg, double screws | 5 | Brackets | $1.13 | $5.65 |
| HFS5-767 Misumi 20x20 extrusion 767mm | 3 | 8020 | $5.29 | $15.87 |
| HFS5-140 Misumi 20x20 extrusion 140mm with holes -a | 3 | 8020 | $3.31 | $9.93 |
| HFS5-2040-200 Misumi 20x40 extrusion 140mm | 2 | 8020 | $3.12 | $6.24 |
| HFS5-53 Misumi 20x20 extrusion 53mm | 3 | 8020 | $3.31 | $9.93 |
| Makerslide 250mm | 1 | 8020 | $8.49 | $8.49 |
| Makerslide 500mm | 2 | 8020 | $13.49 | $26.98 |
| Makerslide 750mm | 2 | 8020 | $18.99 | $37.98 |
| Aluminum plate (2' x 3' x 0.5") | 1 | Machine Bed | $43.80 | $43.80 |
|  |  |  | **Section Total** | **$176.42** |
| **Miscellaneous** |  |  |  |  |
| HTS 171C6 6.25" Curved Stainless Steel College Tweezers | 1 | Tweezers | $4.95 | $4.95 |
| Inventables 25312-17 #12 al spacer 0.625 in 15.875mm | 3 | Spacers | $0.30 | $0.90 |
| Inventables 25312-16 #12 al spacer 0.5 in 12.7mm | 8 | Spacers | $0.30 | $2.39 |
| Inventables 25312-13 #12 al spacer 0.25 in 6.35mm | 2 | Spacers | $0.30 | $0.60 |
| Misumi SMKB5-30 spacer 30x5mm | 8 | Spacers | $2.15 | $17.20 |
| bearing 5mm x 5mm x 16mm | 65 | Bearings | $4.23 | $274.95 |
| 608ZZ flanged bearing | 5 | Bearings | $3.55 | $17.75 |
| RoboDigg gt2 20 pulley 8mm bore | 2 | Pulleys | $1.50 | $3.00 |
| Inventables gt2 20 pulley | 1 | Pulleys | $5.49 | $5.49 |
| Inventables idler wheel | 12 | Idlers | $5.99 | $71.88 |
| V-wheel | 20 | Idlers | $4.42 | $88.40 |
| GT2 timing belt | 10ft | Belt | $0.45 | $4.50 |
| coupler 20x25mm 5mm to 8mm | 1 | Coupler | $4.95 | $4.95 |
| Misumi SCCN8-6 shaft collar | 6 | Shaft collar | $1.69 | $10.14 |
|  |  |  | **Section Total** | **$507.10** |
| **Mounting** |  |  |  |  |
| Y idlers holder plate | 2 | PETG | $9.15 | $18.30 |
| Y bearing plate | 2 | PETG | $6.78 | $13.56 |
| Y pulley side plate | 2 | PETG | $5.86 | $11.72 |
| Y motor mount plate | 1 | PETG | $6.04 | $6.04 |
| Y roller right plate | 1 | PETG | $11.14 | $11.14 |
| Y roller left plate | 1 | PETG | $11.98 | $11.98 |
| X idlers plate | 1 | PETG | $8.82 | $8.82 |
| X motor plate | 1 | PETG | $13.34 | $13.34 |
| Gantry front plate | 1 | PETG | $18.93 | $18.93 |
| Gantry back plate | 1 | PETG | $15.44 | $15.44 |
| Z motor mount plate | 1 | PETG | $12.33 | $12.33 |
| PnP head top plate | 1 | PETG | $10.69 | $10.69 |
| Cable chain support mounts | 2 | PLA | $0.22 | $0.44 |
| Actuator mounting plate | 1 | PLA | $0.42 | $0.42 |
| Camera mount | 1 | PLA | $0.33 | $0.33 |
|  |  |  | **Section Total:** | **$153.48** |
| **Fasteners** |  |  |  |  |
| M3 x 6 | 17 | bag of 50 for $5.75 | $0.12 | $1.96 |
| M3 x 8 | 4 | bag of 50 for $5.76 | $0.12 | $0.46 |
| M3 x 10 | 3 | bag of 50 for $5.96 | $0.12 | $0.36 |
| M3 x 20 | 3 | bag of 50 for $7.65 | $0.15 | $0.46 |
| M3 x 25 | 3 | bag of 25 for $5.25 | $0.21 | $0.63 |
| M3 washer | 42 | bag of 100 for $2.10 | $0.02 | $0.88 |
| M3 extrusion nut | 2 | bag of 10 for $3.69 | $0.37 | $0.74 |
| M3 nut | 12 | bag of 100 for $2.81 | $0.03 | $0.34 |
| M4 x 8 low head | 8 | bag of 100 for $13.60 | $0.14 | $1.09 |
| M4 washer | 8 | bag of 100 for $3.13 | $0.03 | $0.25 |
| M4 extrusion nuts | 8 | bag of 10 for $3.97 | $0.40 | $3.20 |
| M4 nut | 8 | bag of 100 for $1.76 | $0.02 | $0.14 |
| M5 x 8 | 21 | bag of 50 for $6.70 | $0.13 | $2.81 |
| M5 x 10 | 7 | bag of 50 for $4.61 | $0.09 | $0.65 |
| M5 x10 low head | 2 | bag of 10 for $7.67 | $0.77 | $1.53 |
| M5 x 12 | 19 | bag of 50 for $4.72 | $0.09 | $1.79 |
| M5 x 15 | 5 | bag of 50 for $12.32 | $0.25 | $1.23 |
| M5 x 20 | 12 | bag of 50 for $6.03 | $0.12 | $1.45 |
| M5 x 25 | 2 | bag of 50 for $6.45 | $0.13 | $0.26 |
| M5 x 30 | 12 | bag of 50 for $6.73 | $0.13 | $1.62 |
| M5 x 35 | 8 | bag of 50 for $11.67 | $0.23 | $1.87 |
| M5 x 40 | 8 | bag of 10 for $7.78 | $0.78 | $6.22 |
| M5 washer | 225 | bag of 100 for $3.86 | $0.04 | $8.69 |
| M5 nut | 52 | bag of 100 for $2.05 | $0.02 | $1.07 |
| M5 extrusion nut | 44 | bag of 10 for $4.18 | $0.42 | $18.39 |
| Eccentric spacer (standard) | 10 |  | $0.43 | $4.30 |
| Misumi HTJ5 self-tapping screw | 30 |  | $0.64 | $19.20 |
| Robodigg Tr8x8 nut | 1 |  | $9.00 | $9.00 |
|  |  |  | **Section Total:** | **$90.57** |
|  |  |  |  |  |
|  |  |  | **GRAND TOTAL:** | **$1,259.32** |

# Appendices (Author: Devin Bidstrup)

## Appendix A - Specifications

| **Metric** | **Value** | **Unit** |
| --- | --- | --- |
| Time to pack full traveler | TBD | minutes |
| Number of Chips Placed in 8 hrs | TBD | chips |
| X-axis max machine speed | 12240 | mm/min |
| Y-axis max machine speed | 12240 | mm/min |
| Z-axis max machine speed | 800 | mm/min |
| Power draw (motors off) | 1.807 | Watts |
| Power draw (motors on) | 65.633 | Watts |
| Accuracy of Number Based Error Checking | 96 | Percent Correct Prediction |

### Power Calculations:

* Power Supply Efficiency at 12V: 85%
* Current draw (motors off) : 0.128A
* Max current draw (motors on) : 4.649A

## Appendix B – Team Information

**Devin Bidstrup (Computer and Electrical Engineer):** Worked primarily on the configuration and communication with the motor controller board, the wiring of the machine, and the python-based control loop.

**George Kent-Scheller (Computer Engineer):** Worked primarily on the computer vision algorithms to find the containers, the python control loop, the chip isolation contouring, and the final enclosure.

**Justin Melville (Computer Engineer):** Worked on the computer vision algorithms to isolate a chip image and then recognize the individual numbers using a Convolutional Neural Net. Additionally designed the GUI and the communication pipe between the GUI and python control loop.

**Joseph Walsh (Mechanical Engineer):** Worked on the mechanical aspects of the design, namely, the actuator, traveler and clamshell constraints, and the enclosure.

**Paul Stephen Hutchinson Maltaghati (Electrical Engineer):** Worked on the wiring and the computer vision algorithm to isolate individual numbers on the chip images.