

The University of Akron

IdeaExchange@UAkron

Williams Honors College, Honors Research
Projects

The Dr. Gary B. and Pamela S. Williams Honors
College

Spring 2024

Robot-Based 3D Printing

Aaron Hoffman
ajh269@uakron.edu

Follow this and additional works at: https://ideaexchange.uakron.edu/honors_research_projects



Part of the [Artificial Intelligence and Robotics Commons](#), [Control Theory Commons](#), [Other Materials Science and Engineering Commons](#), [Polymer and Organic Materials Commons](#), and the [Software Engineering Commons](#)

Please take a moment to share how this work helps you [through this survey](#). Your feedback will be important as we plan further development of our repository.

Recommended Citation

Hoffman, Aaron, "Robot-Based 3D Printing" (2024). *Williams Honors College, Honors Research Projects*. 1855.

https://ideaexchange.uakron.edu/honors_research_projects/1855

This Dissertation/Thesis is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Williams Honors College, Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.

Robot-Based 3D Printing

Aaron J. Hoffman¹

The University of Akron, Akron, OH, 44325, USA

Details of a large-format 3D printer created to print experimental materials, test multi-axis print techniques, and quickly print large objects. The printer consists of a 7-axis robotic arm and pellet extruder, which are controlled by a PC. Experimental materials such as recycled polymers or carbon-fiber reinforced materials can be easily tested with the pellet format of the extruder. The printer can perform different printing techniques and can be used to experiment with material properties when using these techniques with different polymers. The print surface is around 5 times larger than the average commercial 3D printer, and the robotic arm provides a much higher print height. The pellet extruder is controlled by a custom microcontroller interface that is pre-loaded with all extrusion instructions for the print. This microcontroller is commanded by the robot arm controller to synchronize the movements and print operations. The PC compiles commands using RoboDK software, which combines G-Code from a 3D slicer and transforms it into appropriate robot and extrusion commands.

I. Nomenclature

<i>CNC</i>	=	Computer Numerical Control
<i>GPIO</i>	=	General Purpose Input/Output
<i>HLFB</i>	=	High-Level Feedback
<i>PID</i>	=	Proportional Integral Derivative
<i>SPI</i>	=	Serial Peripheral Interface
<i>VDC</i>	=	Volts Direct Current

II. Introduction

Additive manufacturing, commonly known as 3D printing, is a process that involves taking a digital 3D model and making it into a physical object. This can be done in a variety of ways, most commonly cutting the 3D model into slices and printing layer-by-layer. A variety of materials can be used with 3D printing as well, most commonly polymers such as ABS, PLA, or PETG with melting points well above room temperature. These polymer printers use a heated extruder head to melt small portions of fed plastic and dispense it onto a heated print bed or previous print layer. These plastics are commonly in a spool configuration, which allows the extruder to use a motor and feed itself new material until the spool runs out. The extruder is commonly moved via a gantry system, which provides 3 degrees of freedom.

This project's purpose is to implement methods that allow easy experimentation of different printing techniques than the most used ones described above. Rather than using a spool of filament, this printer utilizes pelletized material, which is much cheaper and easier to make. This also allows a hopper system to feed the extruder with plastic, which can print perpetually without pausing to refill. Using pelletized material removes the need to spool experimental materials, which is a timely process and may result in changing the material's properties. By using a robot arm rather than a gantry system, the extruder head can rotate on top of linear movements. This allows for multi-axis prints that can move along the contour of the print, reducing the need for support material and improving structural properties in the print [1].

¹ Honors Computer Science Student, Undergraduate Research Assistant, University of Akron

To achieve this, an MDPH2 pellet extruder from Massive Dimension, and a SIA10F robotic arm from Yaskawa Motoman were combined to create a 3D printer. The MDPH2 can print 2 pounds per hour of pelletized material. It is fed with material through a hopper on the top, where it uses a feed screw connected to a stepper motor to move the material through the heating elements and out of the nozzle (see Figure 1). The extruder supports a wide range of nozzles, and for this project utilizes a 1.5mm nozzle to create large prints in shorter time but lower resolution than a traditional desktop printer. The extruder heater has an external temperature controller, which allows the user to set a desired temperature point for the extruder.



Figure 1: MDPH2 high-level diagram

The SIA10F robotic arm has a maximum vertical range of around 1.2 meters, and maximum horizontal range of 0.7 meters (see Appendix A). The heated print bed is a 0.6-meter square, around 4 times larger than a traditional print surface. It is controlled via the FS100 controller and its programming pendant. The controller has general purpose input/output (GPIO) capabilities, which allow it to communicate with external devices or controllers. This is utilized to communicate with a custom controller for the MDPH2 extruder.

III. Extruder Controller


A. Extruder Interface

The MDPH2 extruder has a generalized interface for connecting and commanding the extruder (see Appendix B). The interface consists of binary pins, which are HIGH at 5-24VDC. The enable pin functions as a safety that ensures the extruder is only commanded when its state is HIGH. This prevents stray signals or floating pins from unintentionally commanding the extruder. The pulse pin commands the extruder motor to take a step, and the direction pin dictates which direction the motor moves. The HLFb, or high-level feedback, pin is a configurable interface from the motor which can output characteristics of the motor such as velocity, torque, or binary signals whenever the motor reaches the commanded state.

B. Hardware Outline

The processor of the extruder controller is a Teensy 4.1 microcontroller. This board was selected since it has GPIO, digital interrupt, Ethernet compatibility, SD card compatibility, and a 700MHz clock speed, making it capable of

executing extrusion instructions as fast as possible. The GPIO is used to interact with the interface of the extruder as seen above. The GPIO is connected to a bi-directional level shifter board. This shifts the Teensy 3.3VDC logic up to 24VDC logic that is compatible with both the extruder and robot controller and shifts the 24VDC logic to 3.3VDC logic that the Teensy can handle. The pins are broken out into an 8-pin female Molex connector, which can connect directly to the extruder (see Appendix C). Most notably, the Teensy can emulate an analog signal with a digital pin, meaning that setting the speed of the motor can be done as seen in Figure 2 below. This code sets the analog frequency of the pin to the desired speed, then begins the signal with a 50% duty cycle.



```
// Analog frequency to achieve the given rev/s
float analogFrequency = revolutionsPerSecond * INPUT_RESOLUTION;
analogWriteFrequency(PULSE, abs(analogFrequency));
analogWrite(PULSE, 128); // 50% Duty Cycle
```

Figure 2: C++ code to pulse analog pin for desired motor speed.

Since the extruder has an external temperature controller, the temperature set point cannot be changed by the extruder controller. This is an inconvenience when setting up a print since it requires the user to manually set the extruder temperature rather than using G-Code instructions generated by the slicer as most desktop printers do. To solve this, a thermocouple amplifier is a part of the extruder controller, allowing the controller to read the Type K thermocouple (denoted as “K-Probe” in Appendix B) on the extruder head. The amplifier converts the voltage generated by the thermocouple into a temperature reading, which it sends to the Teensy over SPI protocol. The amplifier is read at a rate of 4Hz – or every 250 milliseconds.

The design was first modeled on a breadboard to ensure that connections and assumptions were correct. The breadboarded circuit was documented into a schematic and manufactured as a Printed Circuit Board (PCB) (see Appendix C). The PCB served as a much more stable version of the breadboard version, which would commonly have errors caused by wires being accidentally removed or shorts from frayed wires.

C. Software Outline

The Teensy 4.1 is programmed using C++ and the Arduino IDE. This was chosen due to ease-of-use and a large community, which makes troubleshooting easier. An Arduino program consists of two major parts – the setup and loop. Setup is a function that runs once and is used to initialize the hardware of the microcontroller, including GPIO and communication buses. The loop is a function that runs indefinitely and is used for the operational responsibilities of the microcontroller, such as reading communication lines and operating the extruder.

```

void setup() {
  // Initialize Serial Port for communication
  Serial.begin(9600);

  // Initialize Extruder Pins
  pinMode(ENABLE_PIN, OUTPUT);
  pinMode(PULSE, OUTPUT);
  pinMode(DIRECTION, OUTPUT);
  pinMode(HLFB, INPUT);

  // Initialize Communication Status Led
  pinMode(LEDPIN, OUTPUT);

  // Initialize Ethernet/UDP
  SetupUDP();

  Serial.println("Initialization Complete");
}

void loop() {
  UpdateCommand();
  UpdateExtrusionTime();
}

```

Figure 3: firmwar3.ino setup and loop functions

The controller's responsibilities are broken out into modules that follow the inversion of control design pattern and can be seen in Figure 3 above. The controller has two major responsibilities, which are to control the extruder motor, and control the extruder temperature. Additionally, an ethernet connection is used to communicate with the host PC for diagnostic and debugging purposes.

1. Motor Control

The motors are controlled via the interface described in the hardware outline. Extrusion instructions are the desired extrusion speed and duration, which are calculated during the post-processing the robot commands from the extrusion coefficient and robot speed generated by the slicer. The instructions are preloaded onto the SD card where during startup, the extruder controller reads instructions from the SD card and loads them into RAM for faster access. These values are pre-calculated to reduce the number of operations the Teensy needs to perform while printing.

A motor control instruction is executed when the robot controller pulses a digital pin high. The digital pin on the extruder controller is registered as an interrupt, which is a callback function that generates when an event (HIGH or LOW) happens on a specified pin. When the interrupt is triggered, the extruder controller performs the next instruction by setting the extruder speed and restarting the instruction timer.

2. Temperature Control

The temperature control is implemented as a proportional integral derivative (PID) control loop, the same as the stock controller from the extruder. The difference and reason for implementing it in the custom controller is so temperature control does not need to be manually set by the user and is already generated as a part of the G-Code from the slicer. The controller works by having a desired temperature (set point) and the measurement (feedback) and taking the difference between them. This difference is multiplied by the gains (denoted K_p , K_i , and K_d in Figure 4) then plugged into the respective equations. The proportional error measures how far the set point is from the measurement. The integral accumulates the error over time, to account for steady-state error seen when only using the proportional error. However, this commonly results in an overdamped system that oscillates between being above and below the set point. To solve this, the derivative of the measurement is used to anticipate when the reading reaches the set point and prevent oscillations [2]. The results of each are added together, and the result is the output of the heating element of the extruder. Since the heating element is a binary control, if the result is above 0, the heating element is turned on.

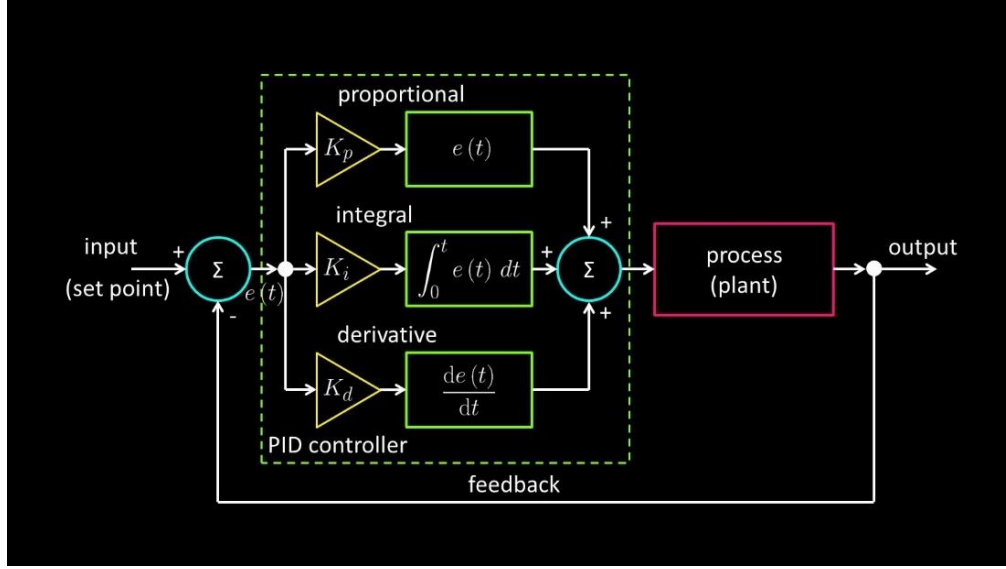


Figure 4: PID controller outline [2]

Since the room temperature of the extruder is much lower than the set point required to melt any plastic, the PID controller demonstrates a phenomenon known as integral windup. This is when the integral component of the controller accumulates a large error and severely overshoots the set point when it is reached [3]. To prevent this, the output is clamped, and the integrator stops accumulating error until the result unclamps. The derivative requires a buffer of readings to counteract sensor noise that may be seen in the system. The system keeps an internal buffer of 12 readings, which causes the derivative to be taken over a timespan of 3 seconds. The derivative is taken from some successive readings in that buffer, and the average is taken from those readings to eliminate noise.

IV. Printing Process Outline

To convert a 3D model into machine code that can be understood by the robot, it is imported into a slicer. A slicer converts the model into instructions called G-Code, which is a language commonly used for CNC machining. The slicer also has settings that dictate how the print should be executed, such as the amount of infill, movement speed, and different printing strategies. These settings are commonly tweaked depending on the desired quality and use-case of the print. For example, if a durable print is required, the infill may be increased to make it more rigid, but if the print is purely aesthetic, the infill will be decreased to save material.

While most desktop 3D printers can interpret G-Code, the FS100 controller cannot. However, it is possible to convert the sliced G-Code into robot instructions, and this is done by using a software called RoboDK. RoboDK is a third-party robot control software that allows simulation and programming of robots like the SIA10F. The user sets up an environment that can accurately simulate the printing environment, the extruder head, and the printing surface

(see Figure 5). From there, the program can convert the sliced G-Code into the proper robot instructions, using the extruder head and printing surface as the points of reference.

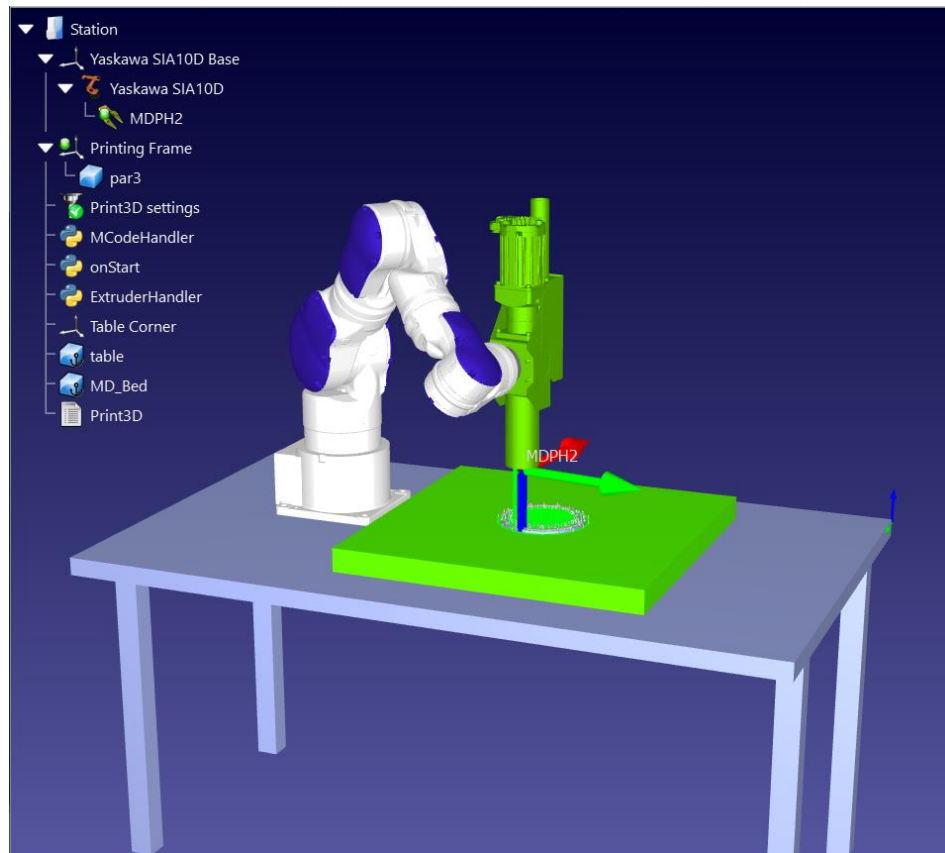


Figure 5: RoboDK workstation

In RoboDK, it is possible to simulate, online program, or offline program the robot with the instructions. Simulating the instructions moves the robot inside of the virtual environment, allowing the user to monitor any strange instructions or collisions, play the print at higher/slower speeds, and see the estimated print time. Online programming executes the instructions concurrently on RoboDK and the physical robot; in this case, executing instructions to communicate with both the robot and extruder controller. This is useful when wanting to run other programs in parallel with the robot instructions. However, due to safety, the instructions briefly pause, and the PC and robot acknowledge each other before performing the next instruction [4]. This makes the online approach impractical for 3D printing since pauses may cause warps or unintentional deposits in the final print (see Figure 6). Offline programming uses a post-processor to convert the RoboDK instructions into a file that can be uploaded directly to the robot. When generating this file, the instructions are run through a post-processor, which ensures that instructions are converted properly and allows customization of the final instruction set. Once these instructions are uploaded, operation is entirely done by using the programming pendant, and RoboDK does not have influence on the instructions. This is the chosen approach since it eliminates the pauses seen during online programming, and safety checks can still be verified via simulating the print. Communicating with the extruder controller is done via the GPIO of the robot

controller, where each movement pulses a GPIO pin that commands the extruder controller to execute the next instruction.



Figure 6: A portion of an online print of a cylinder.

V. Conclusion/Future Plans

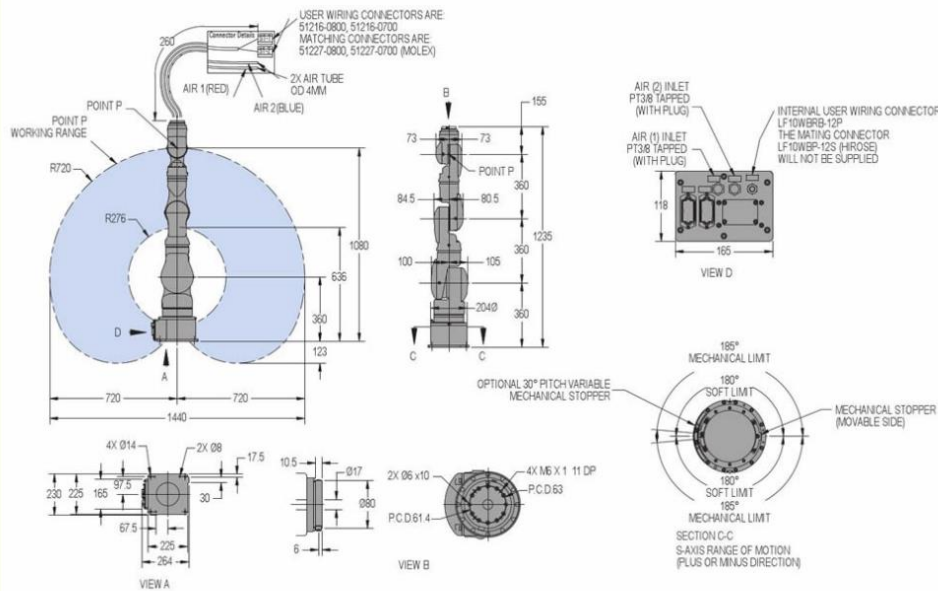
During the system's development, the biggest challenges were those related to the resulting print quality (as demonstrated in Figure 6). The biggest challenge with print quality was making the change from online programming to offline programming, as described in the previous section. Online programming was simple to get a poor but somewhat functional print, but obtaining any quality was impossible due to the safety precautions that RoboDK takes when online programming. Different approaches such as threading the handlers and performance optimizations for the simpler prints were taken, but the online programming was simply too slow for larger, more complex prints. Offline programming then had a different set of technical challenges mostly related to communication between the robot and extruder controllers. However, it naturally solved most of the challenges that were seen when using online programming. In addition, the last performance optimizations for online programming were to preload the instructions onto an SD card, which shared some work with the solution for offline programming as well.

The system created successful visual benchmark prints with PETG material. While these are useful, benchmarking by doing structural tests on standardized prints is the best way to identify the effectiveness and practicality of the system. The ease-of-use of the system also requires some improvement. The current system requires many steps by the user to perform a 3D print between different programs and controllers. Ideally, this would be streamlined into one application that uploads the print via the ethernet connections on both controllers.

Additionally, the goals outlined at the beginning of this document to print with experimental materials and methods have not been fully explored. Experimental materials, including carbon fiber reinforced PLA and mixed recycled materials are planned to be put through the printer and benchmarked against traditional spooling and printing. The team speculates that removing the spooling step changes the structural properties of the material. Multi-axis printing and the effect it has on the resulting prints will also be explored. With RoboDK, the same process can be followed to convert G-Code into robot instructions, which means that multi axis printing only requires the appropriate slicer. This makes approaching multi-axis printing simple, since it is compatible with the existing infrastructure.

VI. Appendices

SIA10F ROBOT



All dimensions are metric (mm) and for reference only.
Request detailed drawings for all design/engineering requirements.

SPECIFICATIONS

Axes	Maximum motion range [°]	Maximum speed [°/sec.]	Allowable moment [N·m]	Allowable moment of inertia [kg·m ²]	Controlled axes	7
S	±180	170	-	-	Maximum payload [kg]	10
L	±110	170	-	-	Repeatability [mm]	±0.1
E	±170	170	-	-	Horizontal reach [mm]	720
U	±135	170	-	-	Vertical reach [mm]	1,203
R	±180	200	31.4	1	Protection - IP rating XP Package (optional)	IP50 Base; IP64 Body; IP67 Wrist
B	±110	200	31.4	1	Weight [kg]	60
T	±180	400	19.6	0.4	Power requirements	1- or 3-phase; 200/230 VAC at 50/60 Hz
					Power rating [kVA]	1.5

OPTIONS

Wide variety of fieldbus cards
Vision systems
Robot base I/O cables
External axis kit
Material handling software package
Conveyor tracking
MotoFit™ force sensing package

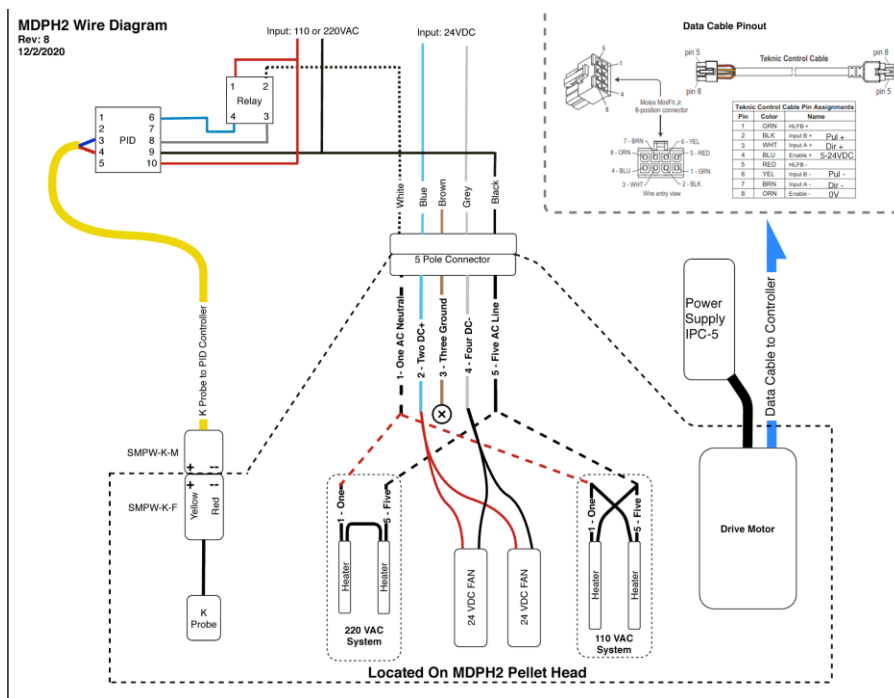


Yaskawa America, Inc.
Motoman Robotics Division

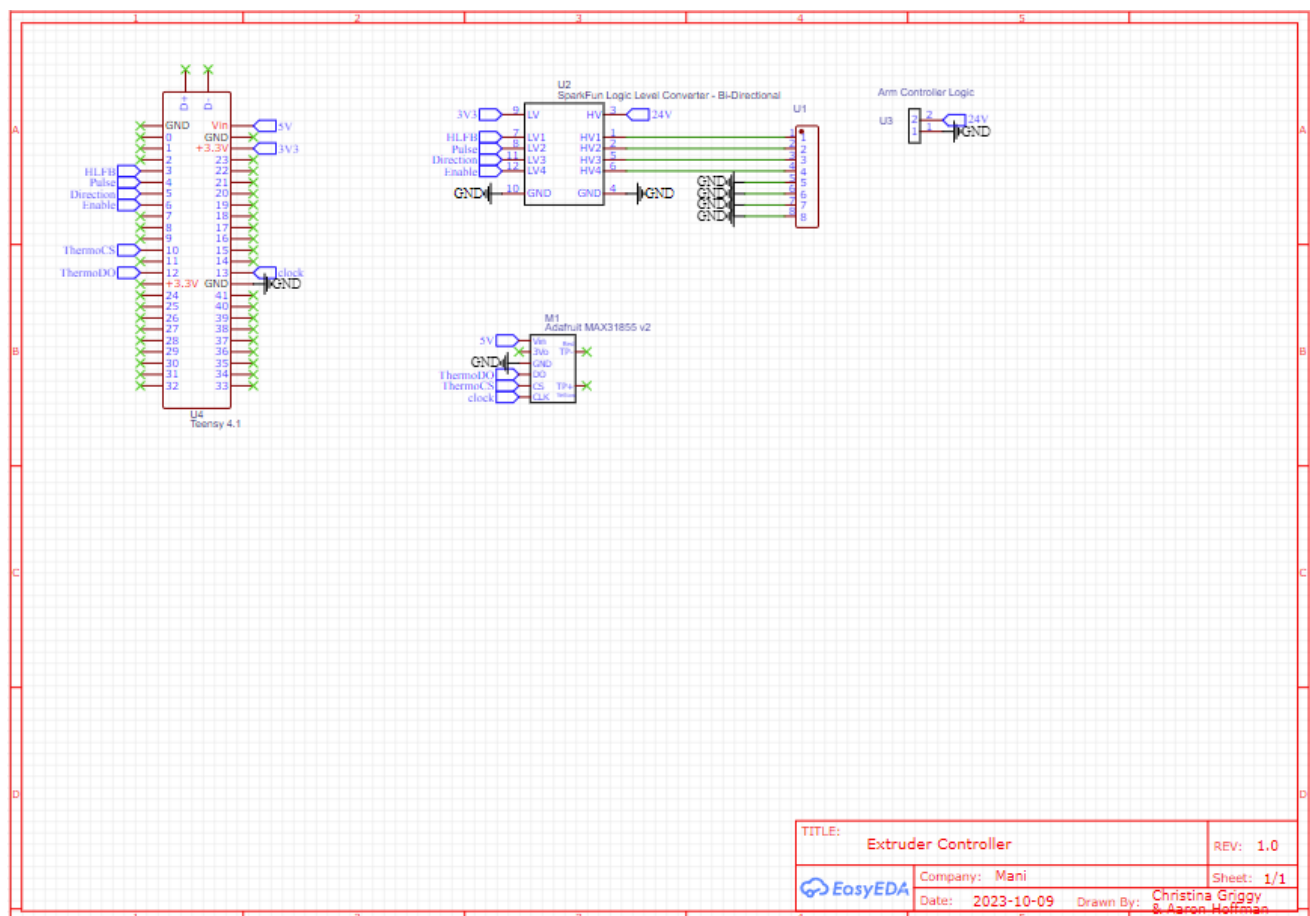
100 Automation Way
Miamisburg, OH 45342
Tel: 937.847.6200
Fax: 937.847.6277

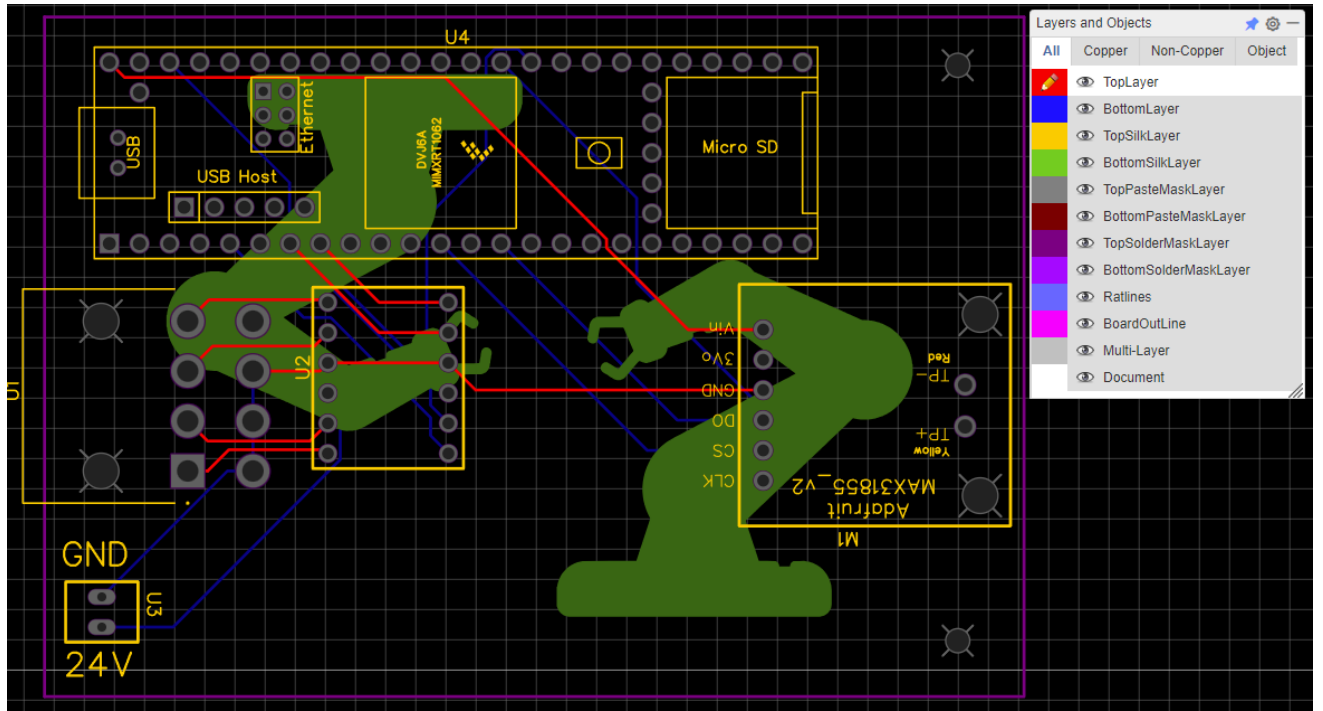
motoman.com

Appendix A: SIA10F robot specifications

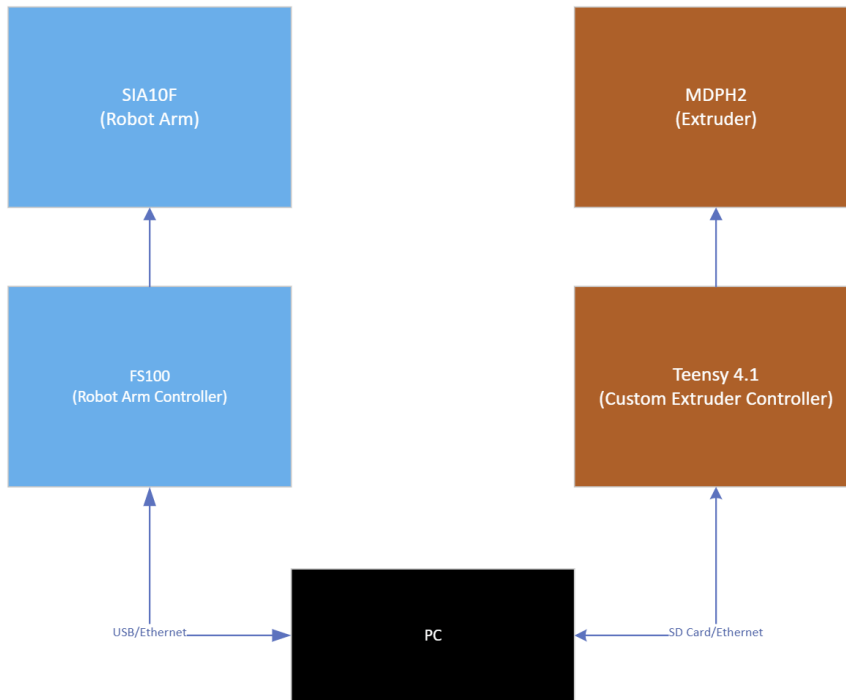


Appendix B: MDPH2 Wiring diagram.





Appendix C: Extruder controller schematic and PCB design



Appendix D: High-level system diagram

VII. Acknowledgments

I would like to thank Dr. Manigandan Kannan for the conceptualization, contributions, and continued support of this project. As well as Christina Griggy and Joshua Slivka for their assistance with various aspects of the project.

VIII. References

- [1] Jan Werner, Mohamed Aburaia, Alexander Raschendorfer, Maximilian Lackner, MeshSlicer: A 3D-Printing software for printing 3D-models with a 6-axis industrial robot, Procedia CIRP, Volume 99, 2021, Pages 110-115, ISSN 2212-8271, <https://doi.org/10.1016/j.procir.2021.03.018>
- [2] ShawnHymel, Introduction to PID Controllers, URL: <https://www.digikey.com/en/maker/projects/introduction-to-pid-controllers/763a6dca352b4f2ba00adde46445ddeb>
- [3] Åström, Karl (2002). Control System Design. pp. 228–231, URL: <https://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom.html>
- [4] SMOOTH MOVEMENT WITH CONSTANT SPEED ISSUE URL: <https://robodk.com/forum/Thread-SMOOTH-MOVEMENT-WITH-CONSTANT-SPEED-ISSUE>