User Manual for Lab-scale Concrete 3D Printer

Building Materials Lab

Indian Institute of Technology Tirupati

भारतीय प्रौद्योगिकी संस्थान तिरुपति

IIT TIRUPATI

Contents

1 Introduction

The lab-scale concrete 3D printer involves a gantry-based system with a mobile bed and a stationary printhead for material deposition. Systems with mobile print beds are uncommon as they have reduced functional areas available for printing compared to systems with stationary print beds. Still, such a system was fabricated to allow for the use of multi-functional printheads (which are heavy, and their actuation is tricky). The superficial controls of the system are written through the MATLAB graphical user interface, which encodes information for Arduino processors through the serial compilation of signals. A schematic of the CAD model of the system and the physically realized system is shown in fig. 1. The fabrication of the system was completed in March of 2024.

(a) CAD model (b) Fabricated system (BM lab-IIT Tirupati)

Figure 1: Design and fabrication of gantry system

1.1 Mechanical system

The motion of the print bed is actuated through stepper motors. The stepper motors are used in applications requiring precise control of actuation. Thus, they find typical applications in 3D printers. In FDM (3D printing of special plastics), stepper motors obtain actuation with errors in the order of $\sim 10^{-2}$ mm. In these systems, the nozzle size is typically around 0.2 mm. Thus, minor errors in a similar order could be fatal. This is not the case in concrete 3D printing, where the width of the deposited layer is at least 10 mm. Thus, precise control of actuation is not emphasised as in the case of FDM.

For the vertical actuation, two stepper motors are connected to lead screws (through couplings), which bear a mounting hub bearing the horizontal frame carrying the print bed. The lead screw allows passive retention of the bed against gravity. The rotation of the lead screw causes slipping of the hub (due to planar constraints as the hub is held against the horizontal frame). Two vertical shafts are equipped for the frame to enhance the rigidity in the vertical plane. These vertical shafts bear a linear bearing which is mounted to the horizontal frame. For the motion in the horizontal plane, two motors on one axis and one motor are equipped with timing belts, translating the stepper motor's rotational motion to the bed's translatory motion. Linear rails (compatible with the 2020 Aluminium rails) are equipped in the horizontal frame and four sliding blocks (2 for each rail) in both perpendicular directions. All the components in the system, barring the rail mounts in the horizontal frame, are available commercially. The remaining elements are fabricated through FDM (their components are made of PLA and are coloured black in the physical system). All components can be easily replaced; the details of the element and suggested retailer are presented in the table below:

1.1.1 Principal assemblies

The description and function of a few select components are presented in this section. The image of a completely assembled system is presented in fig. 9. The principal components include:

Principal assemblies

1. **Lead screw assembly**: The lead screw is employed to carry the weight of the printed material and the horizontal transmission system. The lead screw serves as a passive retention system (can carry weight without the need for an external power supply). The lead screw is a threaded rod, and only constrained rotation of the lead screw causes the hub to slip along the thread, causing vertical motion. The schematic of the lead screw, along with the end components, is shown in the fig. 2. The end of the lead screw is fitted with end supports which carry a mounted bearing and a provision for locking the system using a circlip. One end is equipped with a flexible coupler to allow for leeway for positioning in assembly. The flexible coupler is connected to the stepper motor to rotate the shaft.

2. **Horizontal transmission system**: The horizontal transmission system relies on a timing belt powered by stepper motors for horizontal actuation. The timing belts are fitted on the pulleys and are rigidly attached to the mounts on the rails.

    - **X-transmission system**: A NEMA 17 stepper motor powers the x transmission; the bed is fitted on three (two in fabricated system) linear rails, as shown in the figure below, of which only one is equipped with the belt. The rotation of the motor causes the motion of the belt, which causes the motion of the bed, which is rigidly attached to the bed using three 3D printed holding mounts visible in fig. 3. The motor and the pulley are also held in place through the custom 3D printed parts, which are fitted with threaded inserts for better bonding of the fasteners to the printed material. A schematic of the connection of the pulley to the mobile frame is shown in fig. 3.
    - **Y-transmission system**: The y transmission system is similar to the x-system, but the belt is provided on both sides equipped with synchronous stepper motors; the connections are displayed in the schematic shown in fig. 4

3. **Gantry frames**: The frames are Aluminium 2020, 2040 and 2060 profiles of different lengths connected through corner and straight brackets. The aluminium profile offers space for positioning sliding nuts to serve as fasteners. For bare bolt connections, 8 mm bolts would suffice for M4 sliding nuts. The schematics are presented in fig. 5

4. **Linear rail**: The linear rail effectively reduces the coefficient of friction for horizontal transmission. They are equipped with sliding blocks to which the components are mounted. The effective coefficient of friction between the sliding block and the rail used in the printer is 0.02 (as per the datasheet provided by the retailer); the same shall be assumed for calculation. The rail and the block schematic are presented in fig. 6.

Note 1.1

The linear rail is made of steel (not stainless) and is susceptible to rust, so it must be cleaned and lubricated regularly (once every 6-8 months).

Note 1.2

As the rail is made of steel, it is very rigid and must be accounted for when calculating the composite frames' load-carrying capacity and consequent deflections.

Figure 2: Lead screw assembly

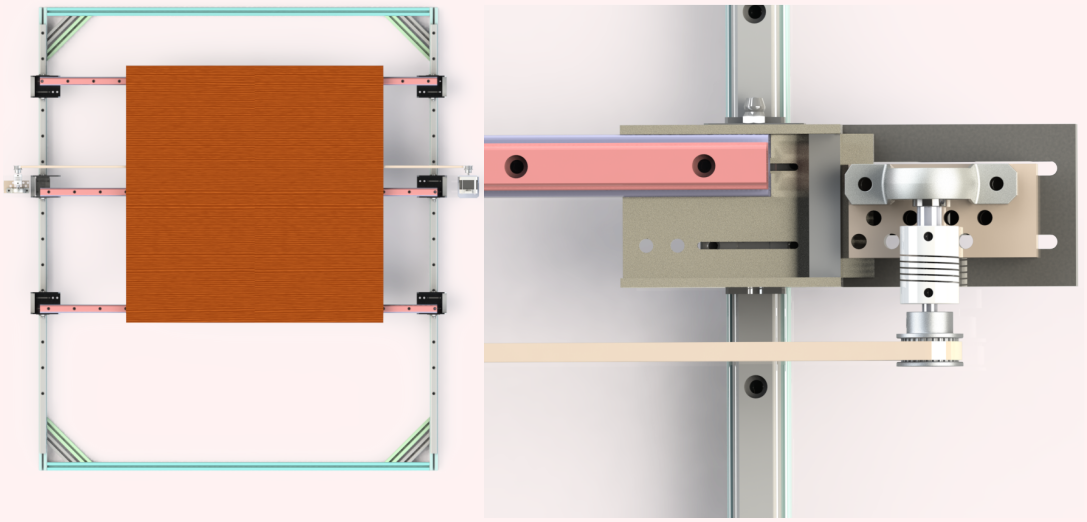


Figure 3: X-transmission system (a) Assembly (b) End fixtures

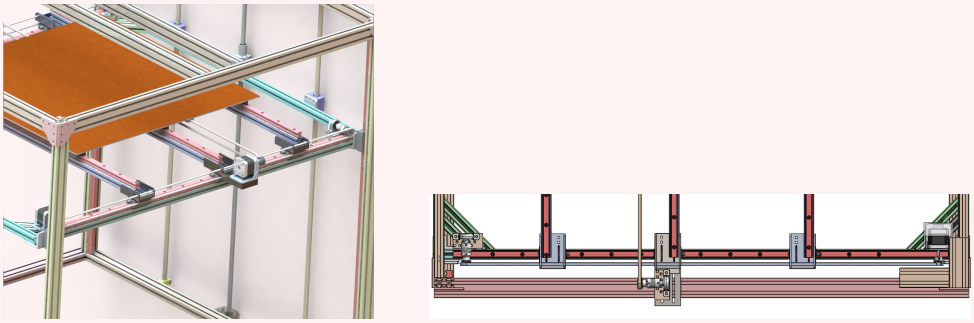


Figure 4: X-transmission system (a) Assembly (b) End fixtures

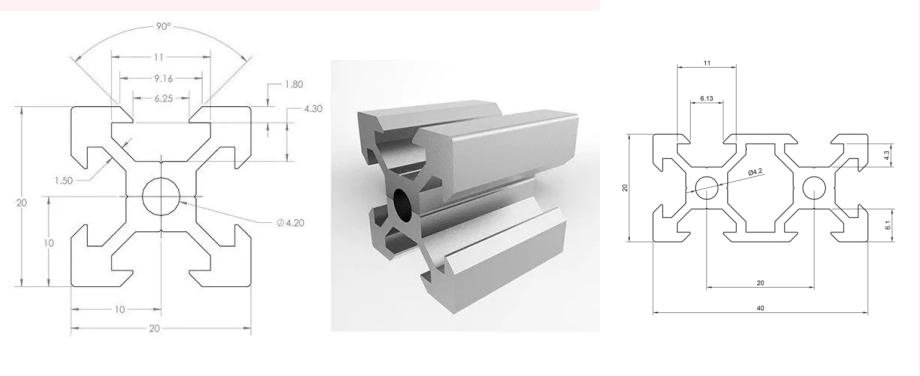


Figure 5: Dimensions of (a) Al2020 (b) Isometric view of Al2020 (c) Al2040 profile

Figure 6: Linear rail with sliding block

1.2 Electronics

The system employs an Arduino UNO and an Arduino MEGA 2560; the Arduino UNO is employed to control the motors in the printhead and the pneumatic pump. The Arduino MEGA is used to control the gantry system. Their circuit diagram is shown in fig. 7. The system uses a TB6600 micro-stepping driver for precise movements, but such precise movements drastically affect the speed of the motor. The micro-stepper has multiple micro-stepping configurations, and it is recommended that a moderate range of micro-steps be employed. The electronics are placed within the gasket, and it is recommended to remain unaltered to ensure the system's controls remain apt. A perforated gasket is used to allow for ventilation of the components (the system may get very hot due to sustained operation; it is recommended that the main terminal be turned off and the Arduino connections be disconnected when not in use- this will prolong the life of the 24V DC power supply). The Arduino boards must be connected to the primary control unit (Laptop) through the USB hub. The MATLAB-Ardinou interface and programming script are explained below:

MATLAB-Arduino interface

A closed loop interface is achieved; the MATLAB GUI receives commands to perform actuation, which transforms it into a digital input for the Arduino interface, which commands the motion of the stepper motor. During the printing operation, the MATLAB interface receives data about the current position of the system and waits till the bed has reached the destination, i.e. commands are *add-on type* (this is done while the printing operation is taking place), whereas, for calibration, the commands are *override type* (i.e. if during transit another command is provided the system will only execute the recent command and neglect past command).

Warning

The closed-loop interface between Arduino and MATLAB might slow the system and can cause delays. Still, these delays are observed to be insignificant for regular 3D concrete printing operations but can be addressed (if needed) through open-loop control by modifying the program appropriately.

For the communication the MATLAB code writes a serial entry of string (For example: A6400B2C3200D1600E0F300G0P) to the device connected in the USB hub, which is read by the Arduino interface, the Arduino program is written in a way that it identifies special characters embedded in the string (i.e. it finds position of letters A, B... in the string), then the program reads the number embedded between these special characters and finds the value (in integer or double format), then the program corresponds these values to certain physical system parameters (the example might correspond to 6400 pulses of motor-1 at a rate of 3200 pulses per second in a particular direction and another motor motion of 1600 pulses at a rate of 300 pulses per second at the direction opposite to the first motor, the last entry provides the information that if any current motion is pending (0-can suggest that the system is currently at rest and 1-can suggest that the system is currently performing another task). Each motor has three input parameters (AxxxxBxCxxxxD): the gap between A and B will tell the number of pulses to be provided to the stepper driver, the gap b/w C and D will tell the rate (rotational speed) at which pulses will be provided, and the gap between B and C will tell the direction 2-corresponding to one direction, 0-corresponding to opposite direction and 1-corresponding to no motion (i.e. A6400B1C3200D will not move). After a command is inputted to the Arduino sensor, the message is overwritten with the same message except that the printing indicator is swapped to 1, all motor direction commands are changed to 1, and the *distance to go* (pending distance to be moved by the bed) is outputted into the MATLAB interface.

Note 1.3

The term pulses refers to digital signals in terms of physical rotational rate. It can be calculated by checking the pulses per rotation metric provided in the stepper motor driver. For example, a motion of 6400 pulses at a rate of 1600 pulses per second at a stepper driver configuration of 3200 pulses per rotation is equivalent to $6400/3200 = 2$ rotations at a rotational speed of $1600 \times 60/3200 = 30$ RPM.

Note 1.4

The stepper driver allows for the provision of changing the pulses required for unit rotation. The TB6600 allows for 6400, 3200, 1600, 800, 400, 200 pulses per rotation. Lower configuration leads to higher rotational speeds at a given digital rate.

Warning

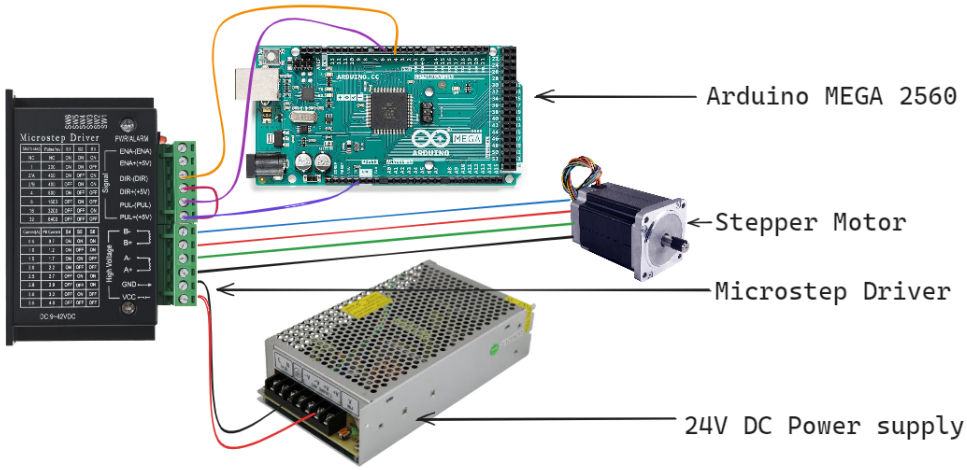
There is a tradeoff between physical speed achieved through lower micro-stepping configurations. It will lead to poor precision.

Note 1.5

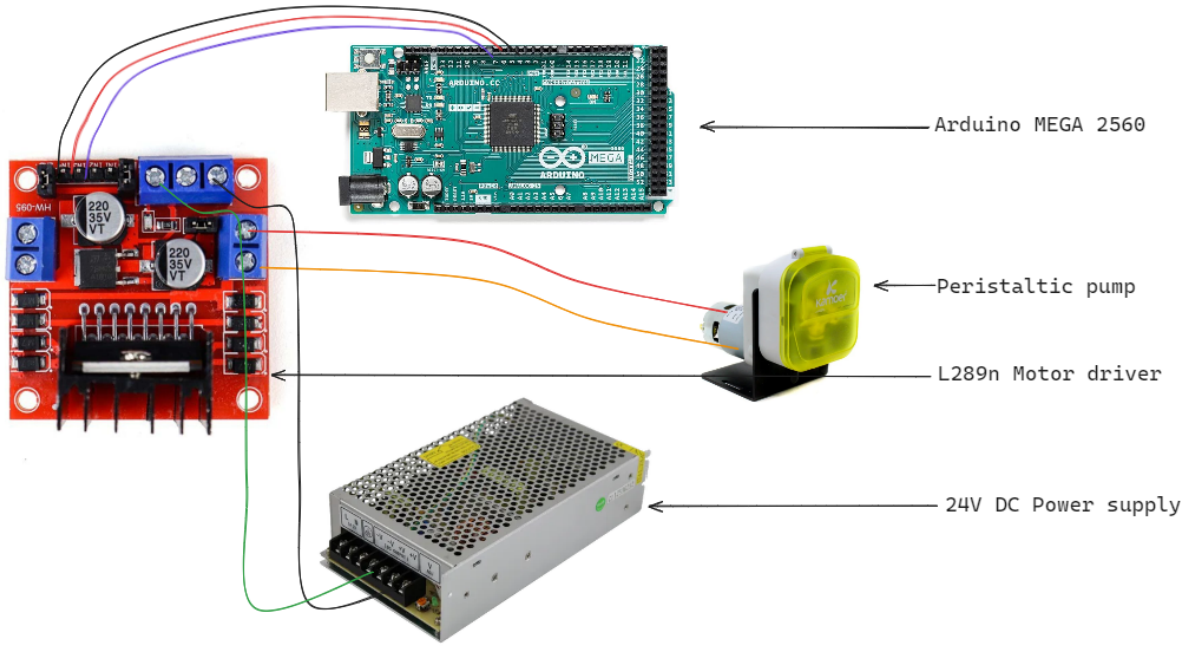
The speed can also be increased by increasing the digital pulse rate through the terminal, but the maximum allowed is 12,800, but stick to values only between 800-6400 pulses per second.

1.3 Graphical user interface

Snapshots of the MATLAB GUI are shown in fig. 8.

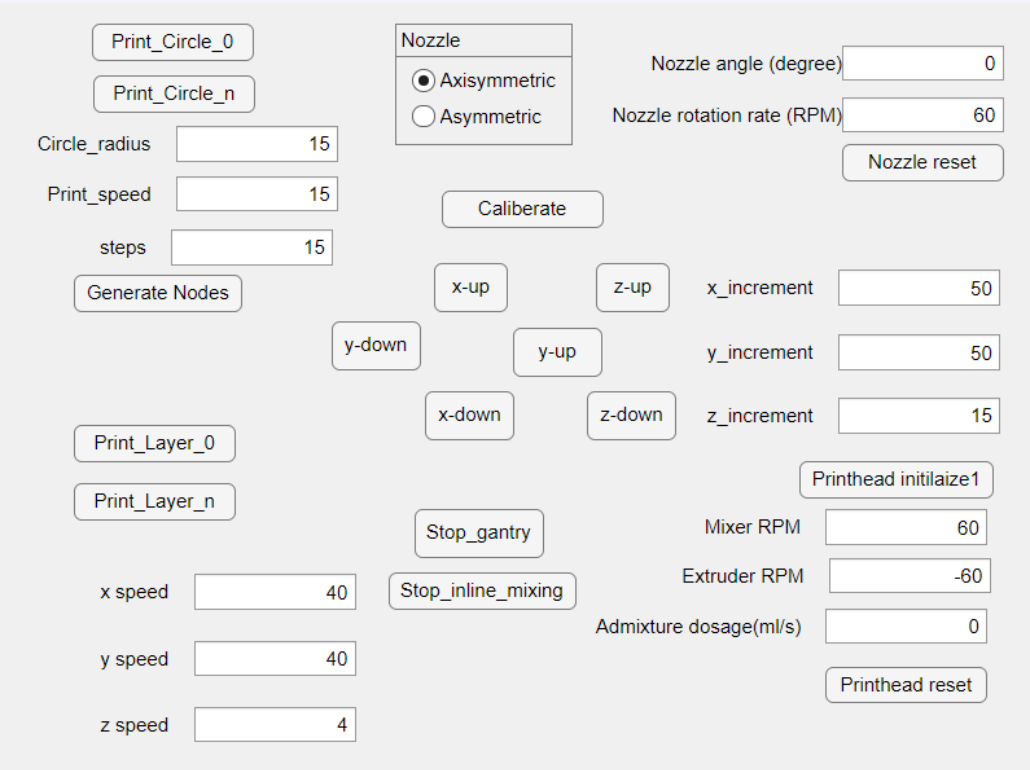


(a) Schematic of circuit assembly for stepper motor

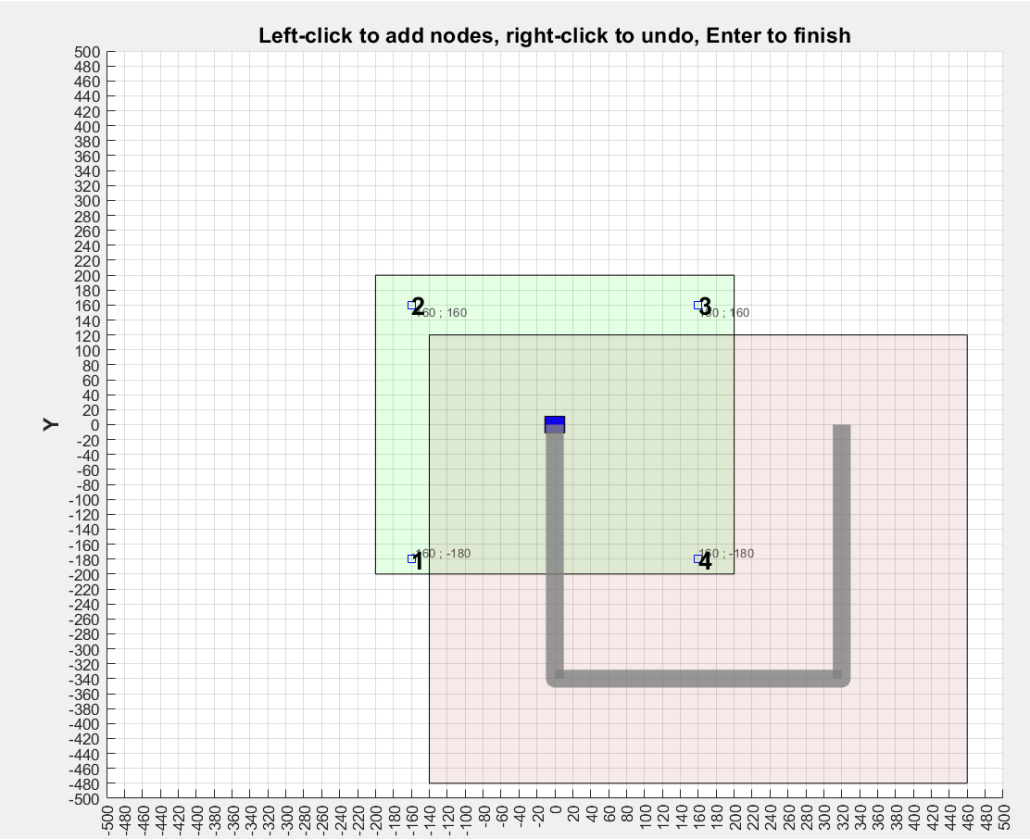


(b) Schematic of circuit assembly for peristaltic pump

Figure 7: Schematic of circuit assembly for actuators involved in the system



(a) Snapshot of graphical user interface used to control the system



(b) Snapshot of a graphical user interface used for assigning coordinates for gantry motion

Figure 8: Graphical user interface for concrete 3D printing

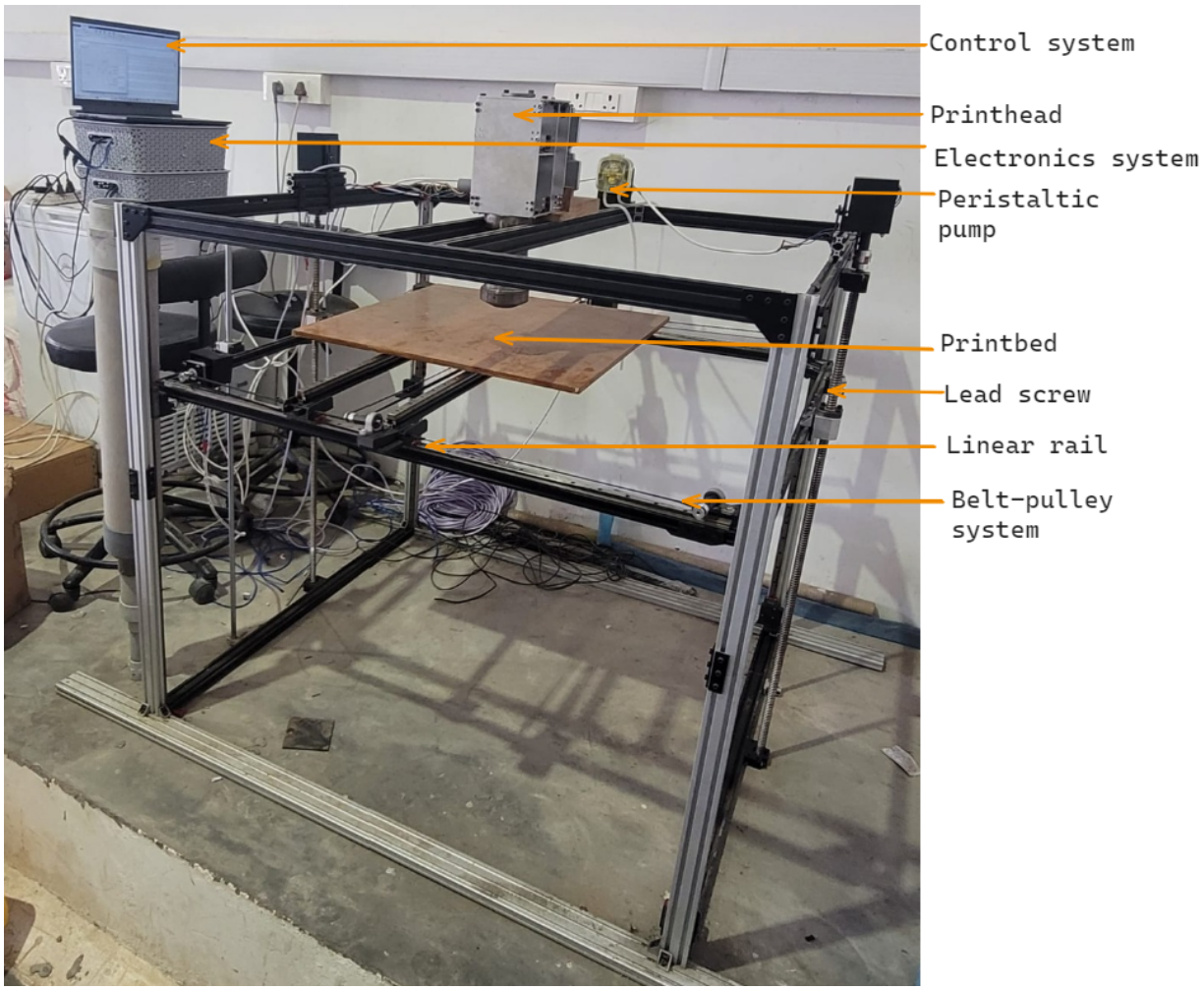


Figure 9: Concrete 3D printer system (Final form)

2 User manual

2.1 Setting-up first time

Procedure

1. Download MATLAB (version later than 2022a) if not already present in the system.
2. In the "Get add-ons" section, search for "Arduino" and download the MATLAB support package for Arduino.
3. Open the folder containing the folders in both MATLAB (all files must be in the same document).
4. Connect the Arduino USB to ports in the laptop/computer. Open the device manager and expand the ports section. Find the COM number for both the MEGA 2560 and UNO board.
5. Open "App.mlapp" in MATLAB, and replace *COM8* by default to the COM number mentioned for MEGA 2560 (use ctrl+F and replace, only one instance has to be changed) and similarly change COM5 by default to value for the UNO board (have to be changed at three instances).
6. Now run the file "App. map" and click on calibrate, upon pressing, you will hear a sudden noise for a few instances. After 10-30s after pressing caliberate try moving the system through the interface.

2.2 Set-up

Procedure

1. Connect the Arduino wires to the USB ports in the laptop.
2. Turn on the main adapter power switch. Upon turning the switch, the micro-stepper drivers will hear a buzzing sound.

   The absence of buzzing noise would suggest wiring issues with the main 24V batteries or their connections to the adapter. Check the external connections.

3. Open and run the MATLAB GUI titled "App.mlapp".

Note 2.1

Upon completion of these steps, a red light should be observed in the Arduino sensor (when the laptop is turned on) and a green light in the stepper motor drivers. Cyan colour lights in the stepper motor driver indicate the connection is interrupted somewhere, and red means that some wiring is disconnected. Refrain from checking it, unless necessary.

2.3 Calibrating/ Manual operation

Procedure

1. Click on the calibrate command in the GUI window. After 10s, press the x-up/down, y-up/down and z-up/down to position the bed manually so that the center of the bed is near the printhead nozzle to maximize print volume.
2. Manually adjust the increment value for the commands if needed (they are set at 15 mm by default). And the speed could also be changed if needed.

   Note 2.2

   The pitch (translator movement for unit rotation) in the horizontal motion is 40.34 mm and 10 mm for vertical motion.

3. For printing with manual commands (for simple geometry such as a wall with a small print volume), this can be achieved through this method.

   Note 2.3

   The commands are override type (i.e. if a command x-up 50 mm is inputted and before the bed has reached the destination (let's say it has travelled 20 mm, and now an x-down 50 mm is inputted, the system will neglect the first command and now move to a final position of -30 mm).

2.4 Using the print-command

Procedure

1. Click on the calibrate button.
2. Position the print bed in the centre through manual operation.
3. Click on the "Generate Nodes" option. A new figure window pops up.

   Generating nodes

   The green area in the figure displays the printable area only on which the cursor can be placed while avoiding contact with the frames. The red area indicates the dimensions of the printed. The blue figure in the centre denotes the nozzle. The grey lining depicts the printed concrete. A representative image is shown in fig. 8.

   3.1 Left-click on the starting point of the print. Upon clicking on the first point, the representative print bed would appear around the point.

   3.2 Left-click on the next destination for the print bed; a consequent printed concrete lining will be displayed in the figure.

   3.3 Continue throughout the trajectory and make sure to close the trajectory.

   3.4 press enter after the last point (preferably the same as the first point).

   Note 2.4

   If the trajectory is not closed, the program automatically closes it with a linear map from the last point to the starting point. It can be modified in software if needed.

Note 2.5

The right button can be used to undo point selection.

4. Prime the printhead, measure the flow and associated extrusion rates, and then set the same value in the speed command field.
5. Once the points are selected and the bed is calibrated, click on the "Print-Layer-0".
6. Once the print of the first layer is complete, manually input z motion through the manual command (modify z-increment as needed) and click on print n layer "Print-Layer-n".

2.5 Prinitng circular columns

Procedure

1. Click on the calibrate button.
2. Position the print bed in the centre through manual operation.
3. Enter appropriate values for the circle radius, extrusion rate and *number of steps*.

   For printing circles, the program discretizes the angular coordinates 0° to 360° into a finite number of points. The program executes linear motion between these finite points, and the speed for the x and y motion along each line is evaluated at the midpoint of lines joining these points.

   Note 2.6

   Ideally, a more significant number of steps would trace a smoother circle. However, do not use values exceeding 100.

4. Click on "Print-Circle-0" for the first layer. item'Click on "Print-Circle-n" for subsequent layers.

2.6 Changing nozzle

Procedure

1. Turn off the power.
2. Using an Allen key, remove the grub screw from both ends of the inserts in the nozzle.
3. Detach the nozzle from the shaft.
4. Select the new nozzle and insert the grub screw
5. Insert the new nozzle in the shaft such that the inserts lie on the D-cut face on the end of the shaft and insert the grub screw in the inserts.

   If the inserts are too outside the nozzle part, use a soldering rod and slowly (carefully, it reaches a temperature exceeding 200°C) push it inside the part.

2.7 Using the rotate-nozzle command

Procedure

1. Switch the nozzle type from axisymmetric to asymmetric
2. In the nozzle angle dialogue box, enter the degree offset you want to provide.

   Note 2.7

   The values are offset from the dynamic position of the nozzle (i.e. command of 30° followed by command of 50° will end up in 80° not 50°)

3. Enter the RPM of the nozzle rotation (60 RPM is the default)
4. Click on nozzle reset

   If the inserts are too outside the nozzle part, use a soldering rod and slowly (carefully, it reaches a temperature exceeding 200°C) push it inside the part.

2.8 Concrete printing process

Procedure

1. Turn the power on and ensure the gantry systems and software function.
2. Using a polystyrene wrap, cover the bed of the printer.
3. Connect the end of the pump hose to the threaded nozzle-inlet.
4. After the material is deposited in the concrete pump's hopper, place a bowl below the nozzle, measure the operational flow rate and derive the extrusion rate.

   Set the extrusion rate as the value for the software's x and y motor speed (mustn't exceed 40 mm/s, and z speed mustn't exceed 10 mm/s). It can be changed based on experimental demands.

5. For the printer's operation, raise the bed to a level layer depth short of the nozzle, and it can be printed using other commands discussed above.

Note 2.8

During operation, to immediately halt the movement press on "stop gantry motion".

2.9 Cleaning (after use)

Procedure

1. After use, stop the pump. ‘
2. Disconnect the motor from the nozzle frame by removing the bolts from the bottom of the plate and also the grub screw connecting the motor and the spur gear.
3. Carefully disconnect the pump hose end from the nozzle

Warning

Be careful with the hose end, as it still carries residual concrete, which can fall on the system.

4. Follow the pump cleaning procedure.
5. Disconnect the nozzle from the frame.

   Do not disconnect the motor wiring; place the detached motor on or near the gantry frames.

6. Clean the hollow shaft and the nozzle using the spray jet.
7. wipe the water on the surface of the parts.
8. Apply oil on the surface (to mitigate corrosion) and wipe the surface.
9. Mount the motor on the nozzle part.
10. Mount the nozzle on the frame once again.
11. Finally, verify the system's controls (software) are working fine.

3 Troubleshooting common issues

3.1 Uneven bed surface

Correction procedure

1. Try manually rotating a lead screw (either) to adjust the level in one direction.

If the bed is not level in the other direction.

1. Loosen the bolts in the vertical direction (on the 4-shafts and 2-lead screw mounting hubs).
2. Use the help of a person to hold the bed in a level position.
3. Tighten the loosen bolts.

3.2 Loose timing belts

The timing belts might loosen due to seasonal temperature fluctuations and the deflection of end supports.

Correction procedure

1. Before disconnecting the timing belts, try tightening the end supports.

   The end supports have two grub screws in the bearing, two bolts mounting them on the frame, and two grub screws on the coupler, and tighten them all.

2. Using a ruler, calculate the length of the bolt needed.
3. If the issue persists, cut somewhere in the middle of the timing belts to remove excess length (on the top part of the belt)
4. Using a strong adhesive, stick the loose ends to the mounts or join the two loose ends using an intermediate timing belt (cut a small strip of timing belt and use it to join both). Refer to the internet for "Tightening loose timing belts" if the procedure is unclear.

3.3 Slipping of lead screws

During the vertical motion, if one of the lead screws is found to lag or halt during rotation (while the other does not), then the connection between the motor and lead screw is loose.

Correction procedure

1. Tighten the bolt between the frame and the motor and the bolts between the motor and motor mount (black steel part to which the motor is held).
2. Tighten the grub screws in the rigid and flexible coupler.

   In the flexible coupler, the bolt connection must be very tight

3. Ensure that the end connections of the lead screw are perpendicular to the horizontal plane.