Lucile Proul

Brianna Moering

Lili Lance

Abigail Hite

Syringe Pump Final Project

Syringe pumps use a motor-driven lead screw which pushes a syringe plunger with precise and controlled motion. As the motor rotates the lead screw, a nut attached to the plunger moves forward, ejecting a proportional and accurate amount of fluid. This is a simple mechanical setup that allows syringe pumps to deliver a wide range of flow rates. It makes them ideal for experiments or systems that require steady, precise fluid control.

To design this syringe pump it must be ensured that the pump is driven by a stepper motor which is connected to a lead screw. The nut must be able to move the plunger forward to dispense and must be compatible with 10 mL and 20 mL syringes. The flow rate must be set by global variables and able to accept decimals in Arduino code. Also, there must be a latch push button (start/stop), and limit switch to indicate when empty. LED lights must be used as indicators (i.e., red = out of fluid, green = running). Wiring must be clean and electrical components must be protected from water exposure, including no holes at the top of the enclosure. All of the components have to be assembled as a single integrated unit that has unique branding of choice. No glue/snap fit can be used as well. The parts are all custom (made by the group) and do not exceed 250 mm to avoid warping. For the software, AccelStepper is used and the flow rate is in steps per second. Additional constraints arose during the building process, such as ensuring alignment between the lead screw and linear rods, managing heat from the stepper motor, designing a syringe holder compatible with multiple sizes, and routing wires to avoid obstruction during carriage movement.

Instructions for Operating Pump

1. Press the Black Button to turn the pump ON. *(Green light ON)*
2. Press the Black Button again to stop automatic pumping. *(Yellow light)*

3. Press Button 2 to move forward (push fluid out). Press the Black button to stop.
4. Press Button 1 to move backward (pull plunger back). Press the Black button to stop.
5. If the pump hits the limit switch, it stops automatically and the light turns RED.

LED Meaning:

- Green = Pump running
- Yellow = Pump stopped
- Red = Safety stop (limit reached)

Design Description

Combining the supplied parts and 3D printed custom parts, the syringe pump design allowed for movement and easy insertion of two syringe sizes. By modeling the pump in Fusion prior to printing (see Figure 1), it was ensured that everything would fit a work together.

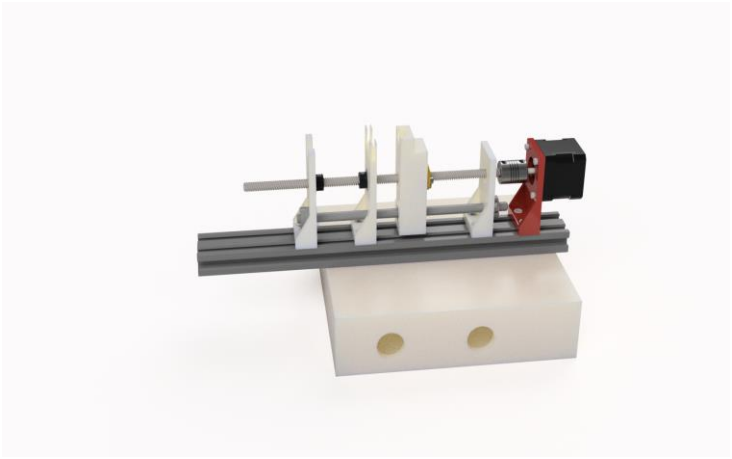

Figure 1: 3D Rendering of Final Syringe Pump Assembly

On the back of the pump, a stepper motor is screwed into a motor mounting plate made out of ABS to ensure stability since the motor has a lot of force and heat. When turned on, the motor rotates a threaded rod, also known as a lead screw. This threaded rod goes through a carriage that has a threaded insert, so when the rod turns, the threaded nut stays in place, moving up the rod. Because the threaded insert is screwed to the carriage, the carriage moves back and forth with the rod. The carriage holds onto the end of the rod so when it moves back and forth, it brings plunger with it, enabling fluid to release of draw in fluid. The diameter of the hole is that of the large syringe, and was made with lips just long enough to reach the diameter of the small syringe to be able to hold both in place. In order for this mechanism to

work, the barrel of the syringe has to remain in place and cannot move with carriage. To hold the barrel in place, there is a support piece that has a slot through the center of the top for the barrel's flange to rest in, preventing forward and backwards movement. This hole does not have sides to it to accommodate both syringe barrel widths. Lastly, to ensure the syringe stays level, one support was added in the front that the syringe loosely rests on. This support has a circular cut out the size of the diameter of the large syringe so that both the large and small syringes can rest on it. The holes for the rod in the supports also have bearings and locking collars to prevent movement of the rod. To stop motion once the syringe has emptied, there is a limit switch attached to the front side of the carriage. Once the carriage reaches the syringe flange, which means the plunger has gone as far as it can, the limit switch comes into contact with the barrel support, activating the switch and signaling the pump to stop.

To automate the motion of the syringe pump, a circuit board that included a microcontroller and various electronic sensors were designed and programmed and contained in a 3D printed box. The syringe pump consists of three buttons (1 latch button & two momentary buttons, one switch, and one LED. The latch button acts as an off and on switch. Once pressed, the speed of the motor is set to zero, and the LED is set to the color yellow to indicate it is at a paused state. The two momentary buttons act as switches to move the coupler in order to control the syringe pump movement to be able to reset the position without touching the coupler by hand. The switch is an auto stop switch for when the syringe reaches its endpoint, and it also is set so that when it's held down, the LED light will be set to red. The LED overall is just an indicator light. It shows when the syringe pump is empty (red), paused (yellow), and running (green). To protect the electronics from finger and water intrusion, a lid for the electronic box was made out of PLA to cover all the electronic parts that could cause damage or be impacted by water contact. As seen in Figure 2, it was ensured no bare wires were exposed.

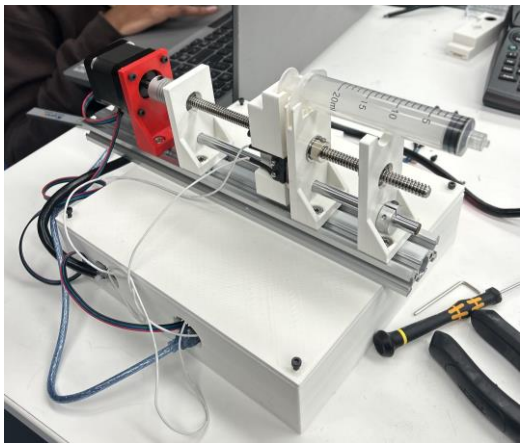

Figure 2: Real-life Assembly of Final Syringe Pump

PLA is non-conductive and has low thermal conductive, which makes it a safe option for housing low power electronic, like the microcontroller, since the outside of the box will not turn hot if the components inside do. In the instance that the lid fails, heat shrinking was also used around any bare wires or soldered joints.

The syringe pump was programmed using the AccelStepper Arduino library to control a stepper motor. The motor was configured to use 1/16 micro stepping, which increases the number of steps per full revolution. The use of micro stepping allows for smooth motion of the leadscrew, reduced vibration. Because micro stepping and flow rates require a fair number of calculations, using variables in the code as described in Table 1 makes it easier to operate and adjust the pump.

Table 1. C++ Variables and their functions

Variable	Meaning/Function
stepPin	Output Pin that sends pulse signals to the stepper driver. Each pulse translates into one motor step.
dirPin	Pin that controls motor direction.
StepsPerRev	Number of full motor steps per 1 revolution
leadScrewPitch	Distance plunger travels per one revolution
syringeDiameter	Used to compute cross sectional area and volume dispersed per mm of plunger movement
flowRate	Desired flow rate (mL/min) converted to linear speed and then steps/sec.
stepsPerMm	Converts linear plunger movement into motor steps
stepsPerSecond	Determines how fast fluid is dispensed
timeInterval	Sets delay between step pulses so the motor runs at the correct flow rate

As described in Table 1, the pumping speed of the motor was set as a variable in order to be able to control the flow rate. The calculated motor speed directly corresponds to the desired volumetric flow rate in mL/min. By adjusting this value in the code, the output flow rate changes. The code also defines the syringe diameter as a variable to make it easily adjustable to account for the different sizes and therefore flowrate of the small and large.

Operating limits of the pump.

The syringe pump has several operating limits that determine how it performs. It can deliver very low flow rates, but it becomes unreliable if the step rate is too slow or too fast. The stepper motor and leadscrew can only apply a limited amount of force as well, meaning the pump cannot push against high resistance. The pumps movement is also limited to the length of the lead screw and the rail base, which restricts how far the syringe plunger can travel. In addition, the motor and driver can heat up during long runs, so extended operation isn't ideal, to prevent overheating.

FFF Printing Issues

There were a few issues with the 3D printing process that were able to be edfix by printing multiple trials so the right fit could be obtained. In the first iteration, when designing the holes for supplied parts to go inside of, specifically in the box for the led lights, in the carriage for the nut screws and the bearing, and in the hole in the barrel support for the linear rod, the dimensions of the exact diameter of the part were used, forgetting to account or tolerances, so the holes were too small. The first iteration of the barrel support is depicted in Figure 3, illustrating an undersized hole of 8 millimeters, the exact diameter of the linear rod as measured in Fusion.

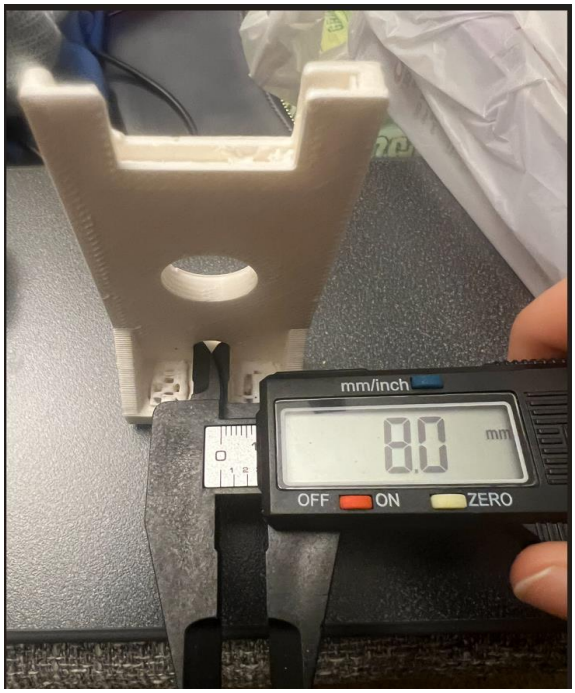

Figure 3: First iteration of Barrel Support With Undersized Hole

Multiple trails with different tolerances were printed, the final one being +0.2 millimeters so the pieces could have a perfect transition fit in the holes. Additionally, while attempting to print the second iteration of the syringe pump components, there was trouble getting the filament to stick to the print bed. More hairspray was added to the surface to increase the adhesive and changing the filament from PLA to PETG, however, it was not successful. It was realized that the printer that was being used did not have doors, which was disrupting the normal airflow, so the printer was switched. It then printed without any errors. Another significant challenge faced was making sure the syringe was level. In the first iteration, all the supports were made to be level with each other. However, because each support met the syringe at a different height, they needed to adjust each support individually to be the proper height to ensure the syringe is straight. For instance, the circular hole in the top of the end support needed to wrap around the diameter of the barrel, while the syringe support had to be even shorter so the flange, which is slightly larger than the diameter, could rest in it. The carriage had its own height since it is not touching the ground and the syringe end has its own diameter.

Example Flowrate Using .1 mL/Min And 20 mL Syringe

$.1\frac{mL}{min} \times 1000\frac{mm^3}{mL} = 100\ \frac{mm^3}{min}$ [Convert Flow To $\frac{mm^3}{min}$]

$$R = \frac{19}{2} = 9.5\ mm$$

$A = \pi r^2 = \pi(9.5)^2 = 283.53\ mm^2$ [Calculate Cross Sectional Area]

$V_{rev} = A \times L = 283.53 \times 2.0 = 567.06\ \frac{mm^3}{rev}$ [Calculate Volume Per Revolution]

$RPM = \frac{100}{567.06} = 0.17641\ \frac{rev}{min}$ [Calculate Required RPM For Flowrate]

$SPR = 200 \times 16 = 3200\ \frac{steps}{rev}$ [Convert To Steps]

$$\frac{steps}{min} = 0.17641 \times 3200 = 564.51$$

$$\frac{steps}{sec} = \frac{564.51}{60} = 9.41\ \frac{steps}{sec}$$

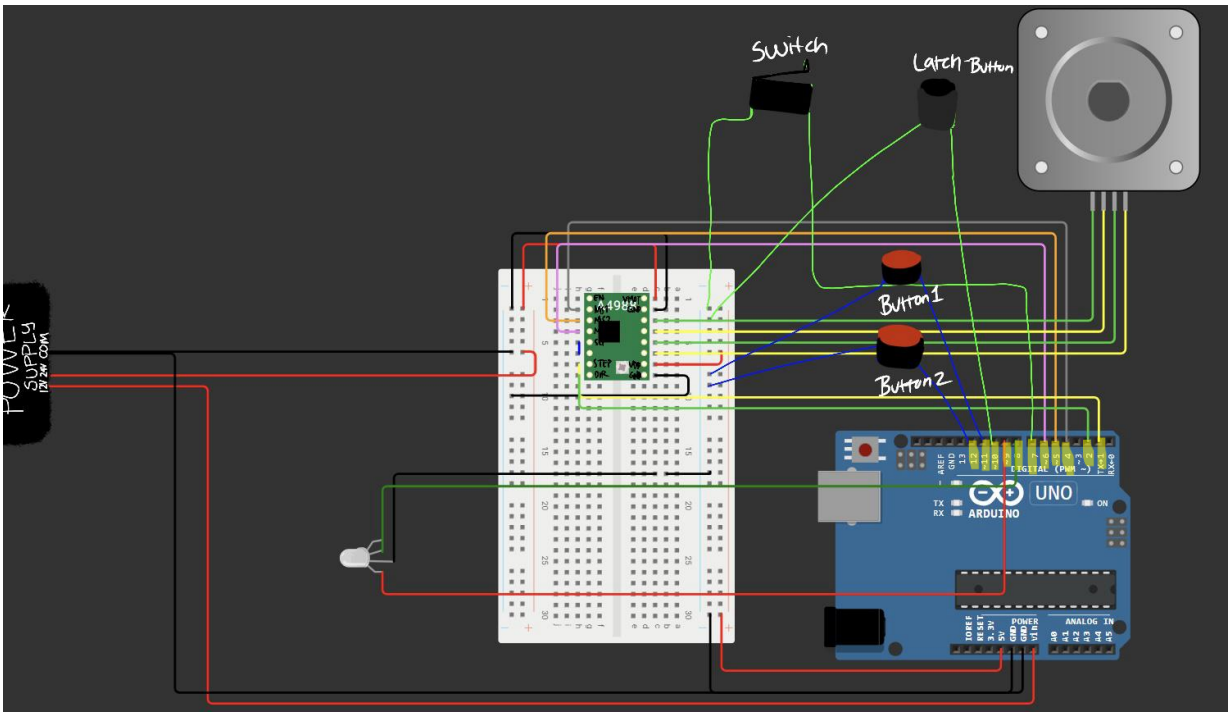

Figure 4: Wiring Diagram

Appendix

Syringe Pump Code

```
#include <AccelStepper.h>

// Configuration
float FLOW_RATE_ML_MIN = .1;
float SYRINGE_DIAMETER_MM = 19;
//float SYRINGE_DIAMETER_MM = 15;
const int MOTOR_STEPS_PER_REV = 200;
const int MICROSTEPPING = 16;
const float LEADSCREW_MM_PER_REV = 2.0;

// Pin definitions
#define STEP_PIN 1
#define DIR_PIN 2
#define MS1_PIN 4
#define MS2_PIN 5
#define MS3_PIN 6
#define LATCH_BUTTON 7
#define LED_GREEN 8
#define LED_RED 9
#define LIMIT_SWITCH 10
#define BUTTON1 11
#define BUTTON2 12

AccelStepper stepper(AccelStepper::DRIVER, STEP_PIN, DIR_PIN);

// State variables
bool motorLatched = false;
bool driveCW = false;
bool driveCCW = false;

// Button states
int lastLatchState = HIGH;
int lastB1State = HIGH;
int lastB2State = HIGH;

// Direction tracking
enum Direction {NONE, CW, CCW};
Direction lastDirection = NONE;

// Debounce timing
unsigned long lastLatchMillis = 0;
unsigned long lastB1Millis = 0;
```

```
unsigned long lastB2Millis = 0;
const unsigned long debounceDelay = 50;

float STEP_RATE = 0;

void setColor(int redValue, int greenValue) {
  analogWrite(LED_RED, redValue);
  analogWrite(LED_GREEN, greenValue);
}

void setup() {
  // Microstepping configuration
  pinMode(MS1_PIN, OUTPUT);
  pinMode(MS2_PIN, OUTPUT);
  pinMode(MS3_PIN, OUTPUT);
  digitalWrite(MS1_PIN, HIGH);
  digitalWrite(MS2_PIN, HIGH);
  digitalWrite(MS3_PIN, HIGH);

  // Motor pins
  pinMode(STEP_PIN, OUTPUT);
  pinMode(DIR_PIN, OUTPUT);

  // Input pins with pullups
  pinMode(LATCH_BUTTON, INPUT_PULLUP);
  pinMode(LIMIT_SWITCH, INPUT_PULLUP);
  pinMode(BUTTON1, INPUT_PULLUP);
  pinMode(BUTTON2, INPUT_PULLUP);

  // LED outputs
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_RED, OUTPUT);

  stepper.setMaxSpeed(1000);

  // Compute step rate for desired flow
  float radius = SYRINGE_DIAMETER_MM * 0.5;
  float area_mm2 = PI * radius * radius;
  float VPR = area_mm2 * LEADSCREW_MM_PER_REV;
  float flow_mm3_min = FLOW_RATE_ML_MIN * 1000.0;
  float RPM = flow_mm3_min / VPR;
  float SPR = MOTOR_STEPS_PER_REV * MICROSTEPPING;
  STEP_RATE = (RPM * SPR) / 60.0;

  stepper.setSpeed(0);
```

```
  // Initial LED state (yellow = idle/paused)
  setColor(255, 255);
}

void loop() {
  int currLatch = digitalRead(LATCH_BUTTON);
  int currB1 = digitalRead(BUTTON1);
  int currB2 = digitalRead(BUTTON2);
  bool limitActive = (digitalRead(LIMIT_SWITCH) == HIGH);

  // ===== LATCH BUTTON (with debounce) =====
  if (currLatch == LOW && lastLatchState == HIGH &&
      millis() - lastLatchMillis > debounceDelay) {

    // Check if motor is currently running in any mode
    bool motorCurrentlyRunning = (motorLatched || driveCW || driveCCW);

    if (motorCurrentlyRunning) {
      // Motor is running - turn everything OFF
      motorLatched = false;
      driveCW = false;
      driveCCW = false;
      stepper.setSpeed(0);
    } else {
      // Motor is stopped - turn latch mode ON
      motorLatched = true;
      driveCW = false;
      driveCCW = false;
      // If no direction set, default to CW
      if (lastDirection == NONE) {
        lastDirection = CW;
      }
    }

    lastLatchMillis = millis();
  }
  lastLatchState = currLatch;

  // ===== CW BUTTON - Always turns ON CW mode =====
  if (currB1 == LOW && lastB1State == HIGH &&
      millis() - lastB1Millis > debounceDelay) {

    driveCW = true;
    driveCCW = false;      // Turn off CCW if it was on
    motorLatched = false;  // Exit latch mode
    lastDirection = CW;
```

```
  lastB1Millis = millis();
}
lastB1State = currB1;

// ===== CCW BUTTON - Always turns ON CCW mode =====
if (currB2 == LOW && lastB2State == HIGH &&
    millis() - lastB2Millis > debounceDelay) {

  driveCCW = true;
  driveCW = false;       // Turn off CW if it was on
  motorLatched = false;  // Exit latch mode
  lastDirection = CCW;

  lastB2Millis = millis();
}
lastB2State = currB2;

// ===== MOTOR CONTROL LOGIC =====
bool motorRunning = false;
bool blockedByLimit = false;

if (motorLatched) {
  // Latched mode - continuous operation
  if (lastDirection == CW) {
    if (limitActive) {
      stepper.setSpeed(0);
      blockedByLimit = true;
    } else {
      stepper.setSpeed(STEP_RATE);
      motorRunning = true;
    }
  } else if (lastDirection == CCW) {
    stepper.setSpeed(-STEP_RATE);
    motorRunning = true;
  } else {
    stepper.setSpeed(0);
  }
} else if (driveCW) {
  // Manual CW mode
  if (limitActive) {
    stepper.setSpeed(0);
    blockedByLimit = true;
  } else {
    stepper.setSpeed(STEP_RATE);
    motorRunning = true;
```

```
    }
  } else if (driveCCW) {
    // Manual CCW mode
    stepper.setSpeed(-STEP_RATE);
    motorRunning = true;
  } else {
    // Idle state
    stepper.setSpeed(0);
  }

  // ===== LED FEEDBACK =====
  if (blockedByLimit) {
    setColor(255, 0);      // RED = stopped by limit switch
  } else if (motorRunning) {
    setColor(0, 255);      // GREEN = motor running
  } else {
    setColor(255, 255);    // YELLOW = paused/idle
  }

  stepper.runSpeed();
}
```

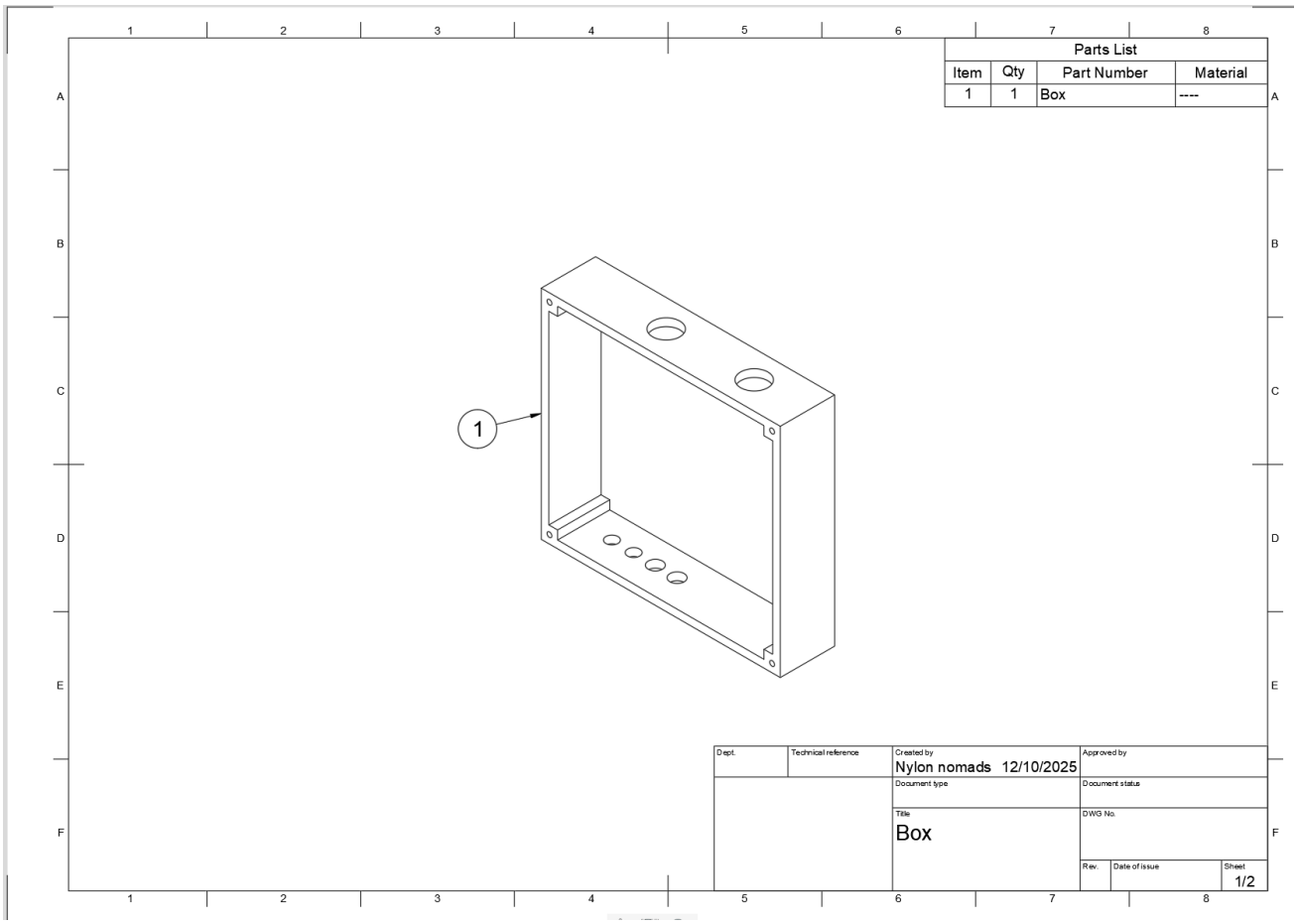

Figure 5: Drawing of Design of Electrical Box in Fusion

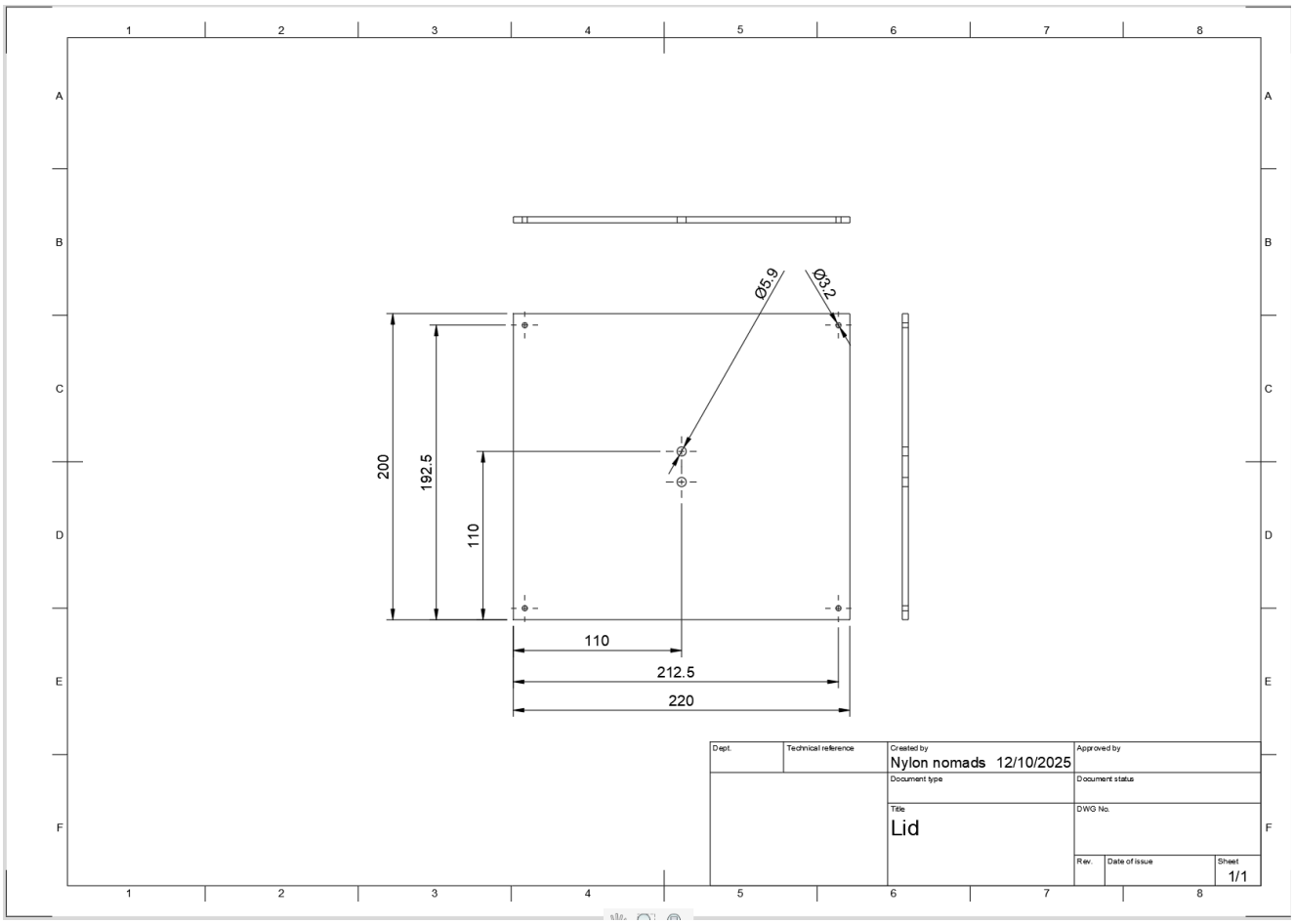

Figure 6: Drawing of Design of Electrical Box Lid in Fusion

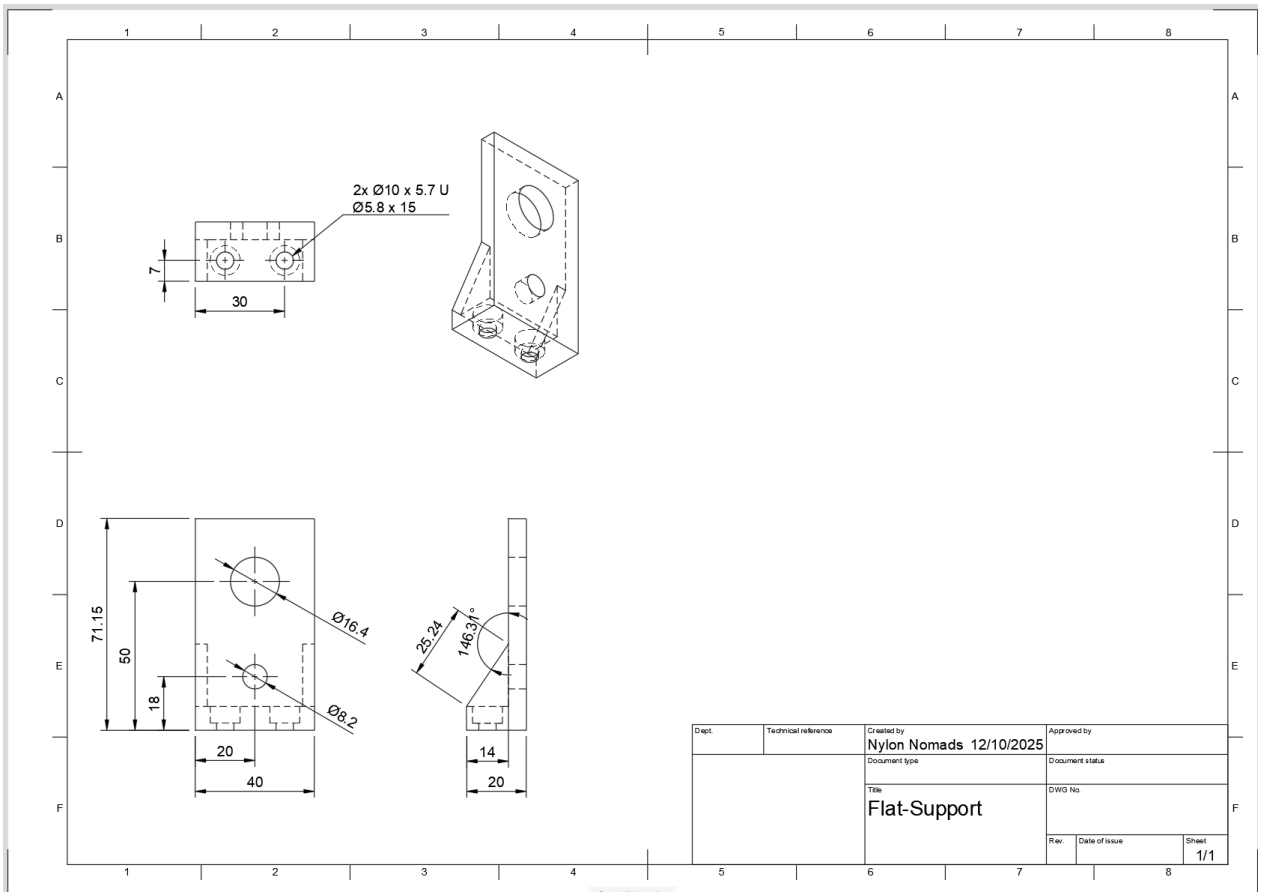

Figure 7: Drawing of Design of Flat Support in Fusion

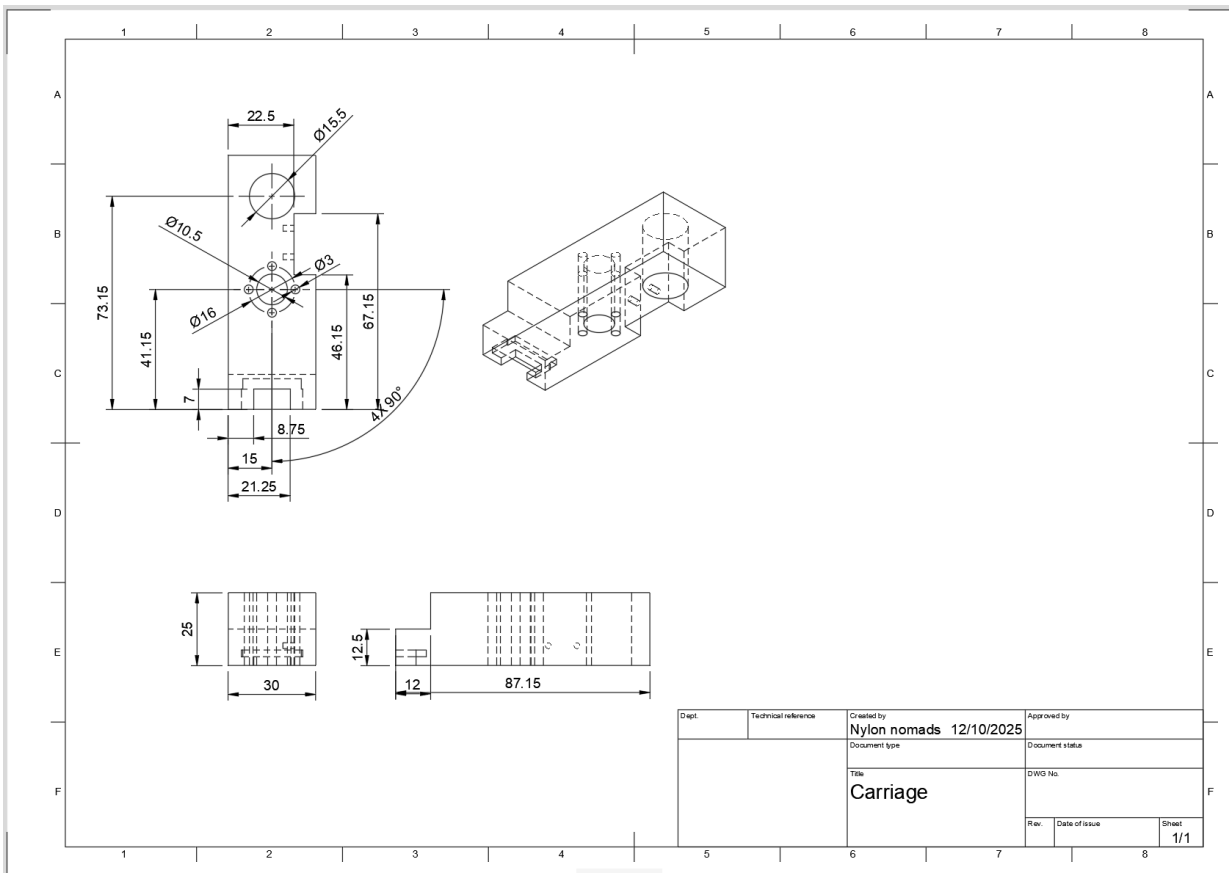

Figure 8: Drawing of Design of Carriage in Fusion

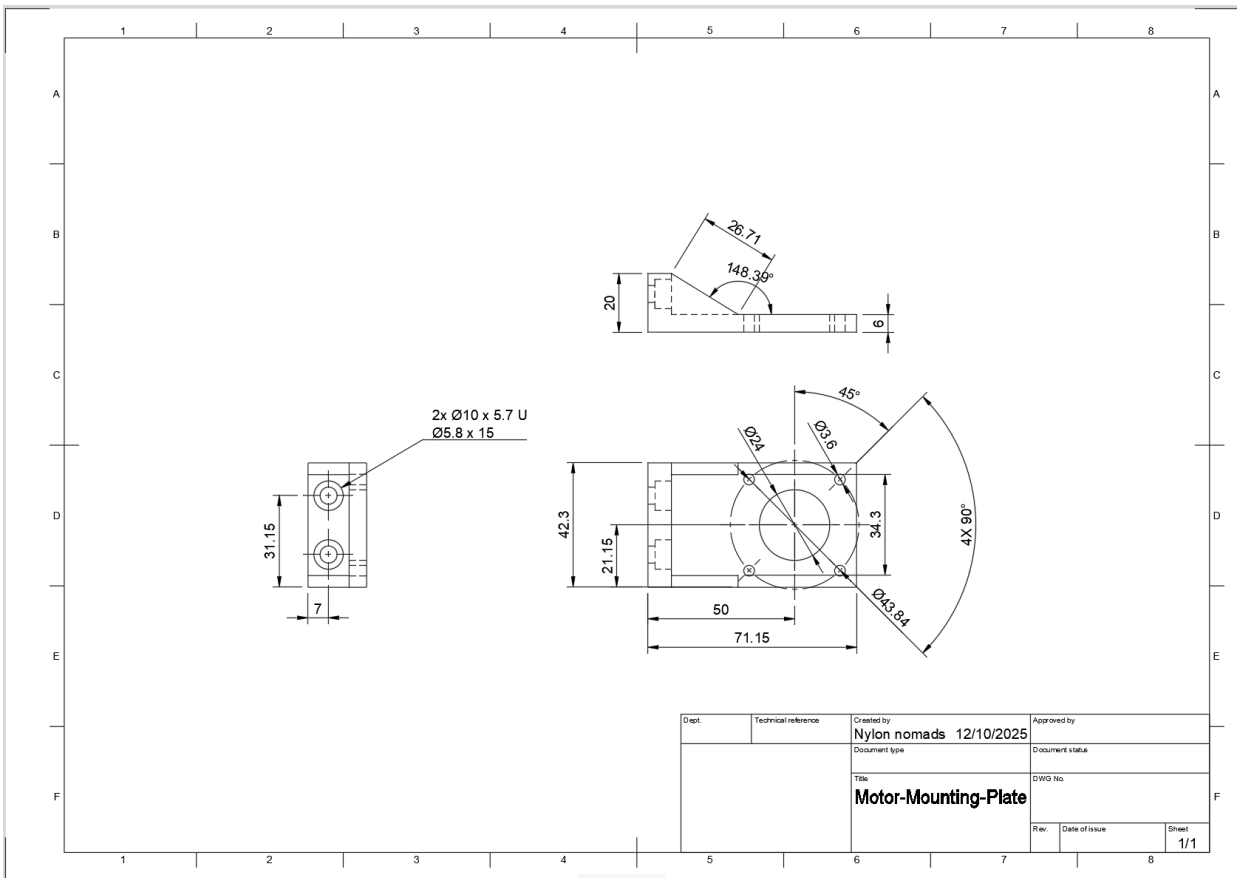

Figure 9: Drawing of Design of Motor Mounting Plate in Fusion

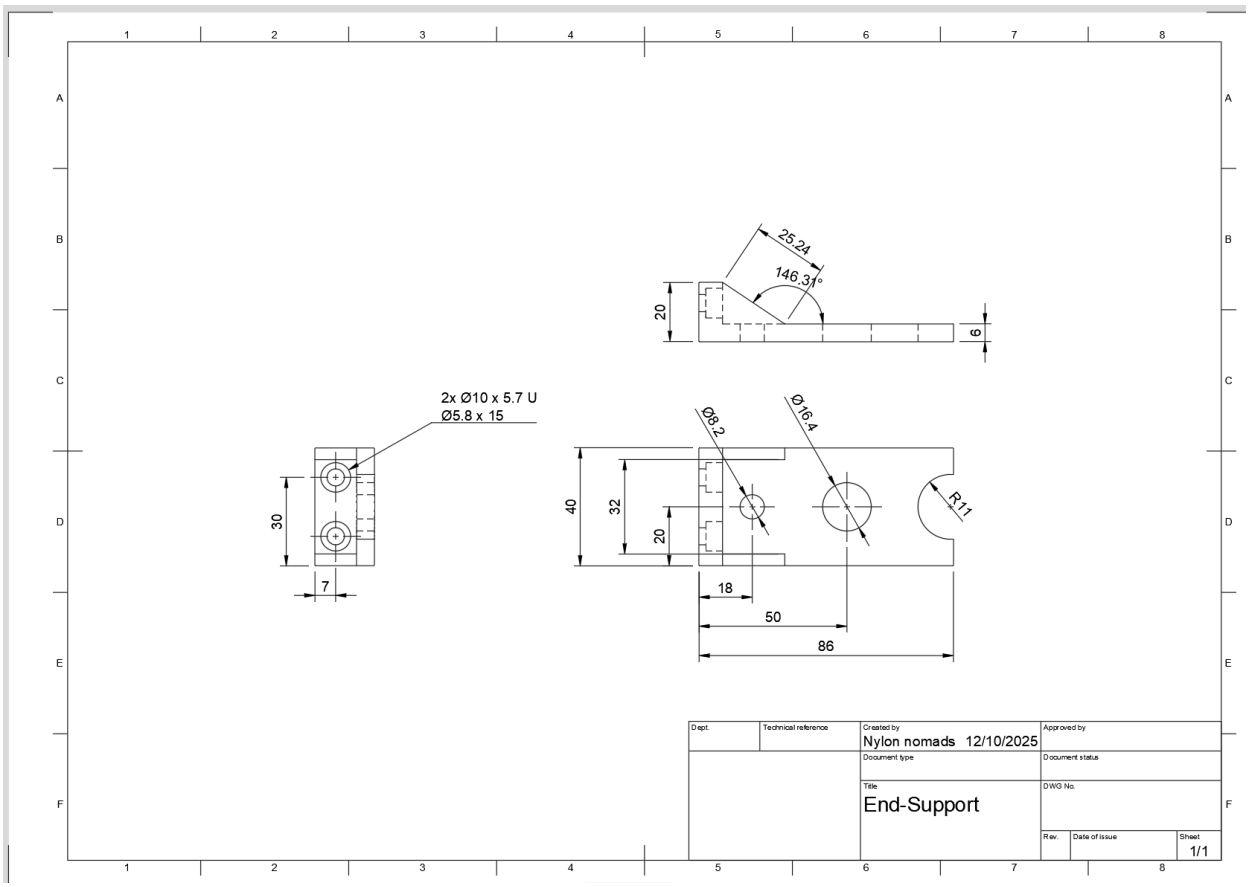

Figure 10: Drawing of Design of End Support in Fusion

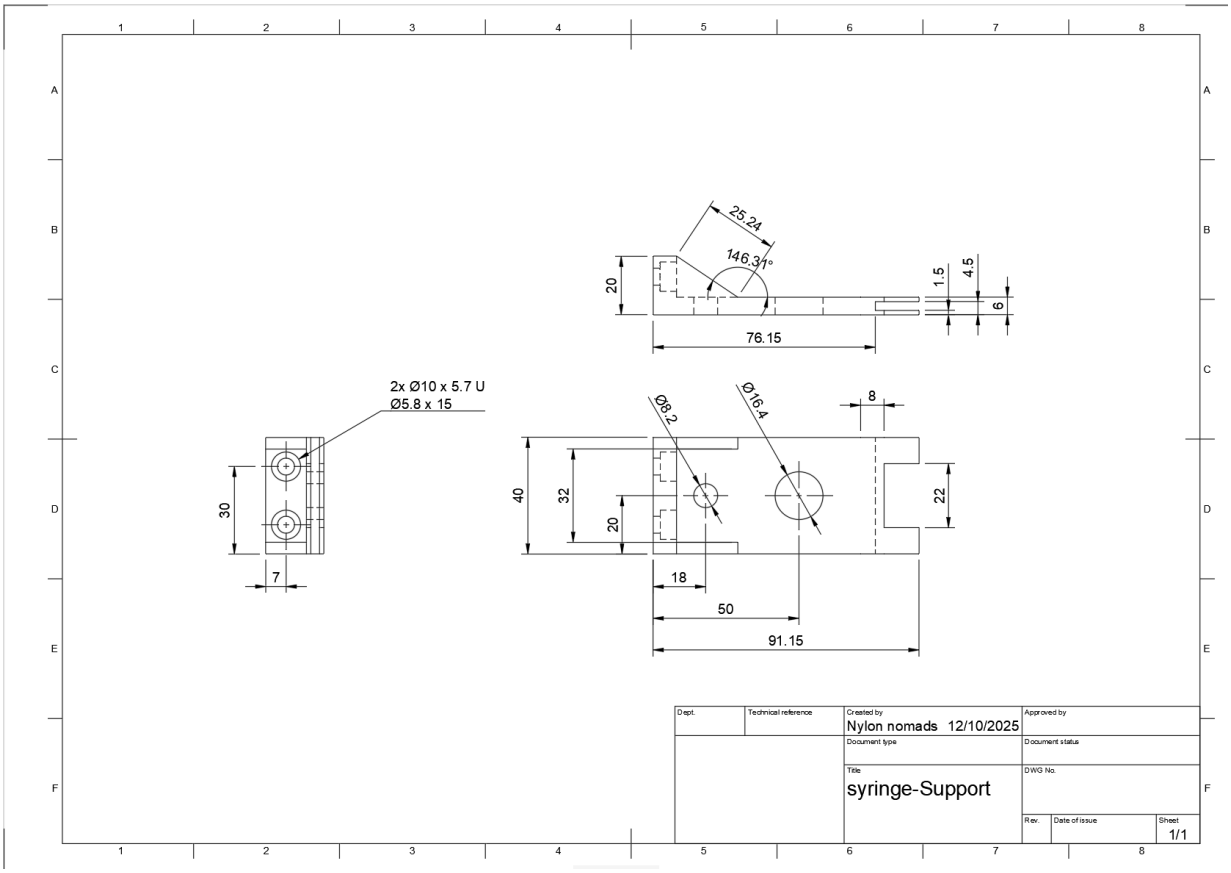

Figure 11: Drawing of Design of Syringe Support in Fusion