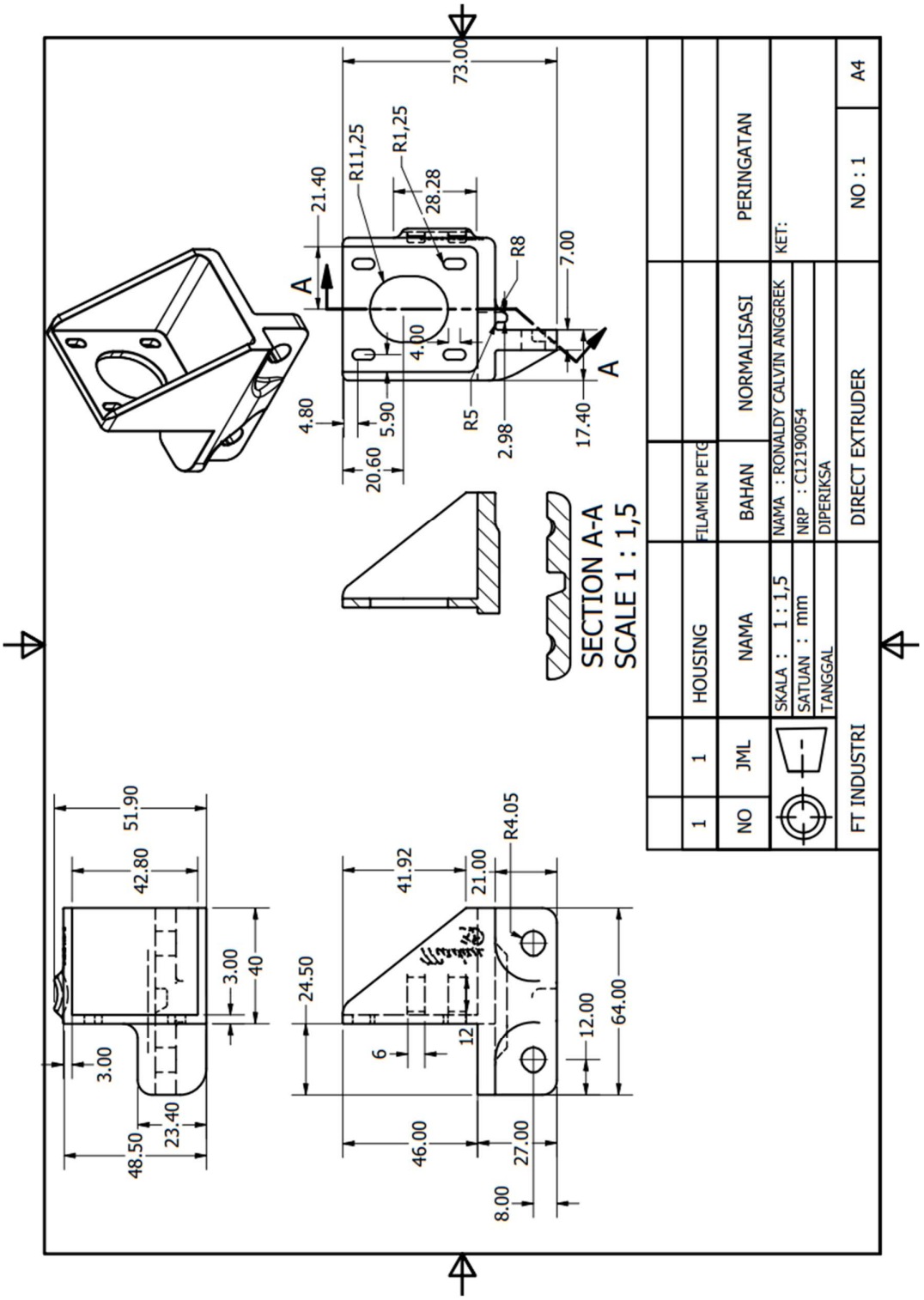
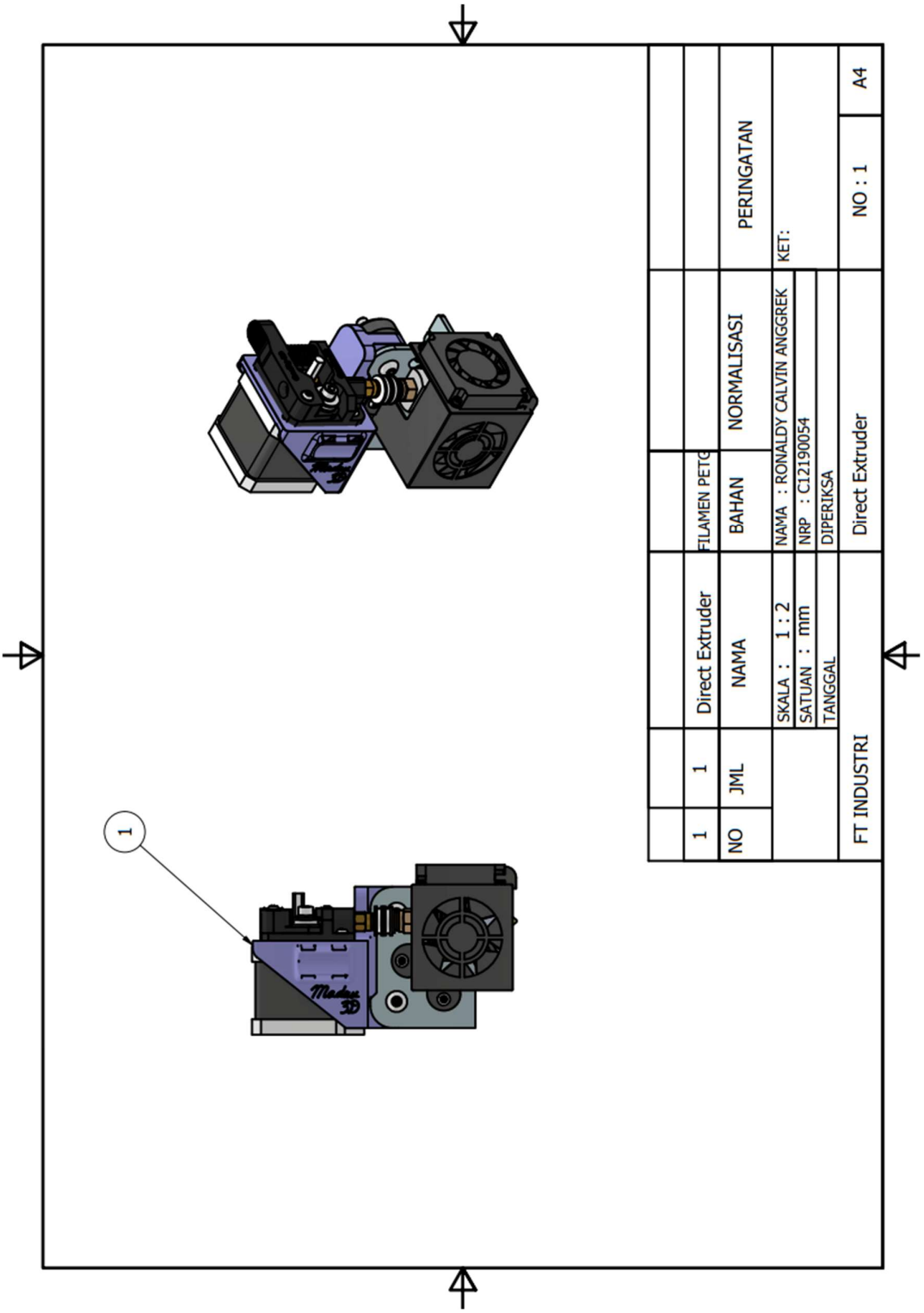
LAMPIRAN

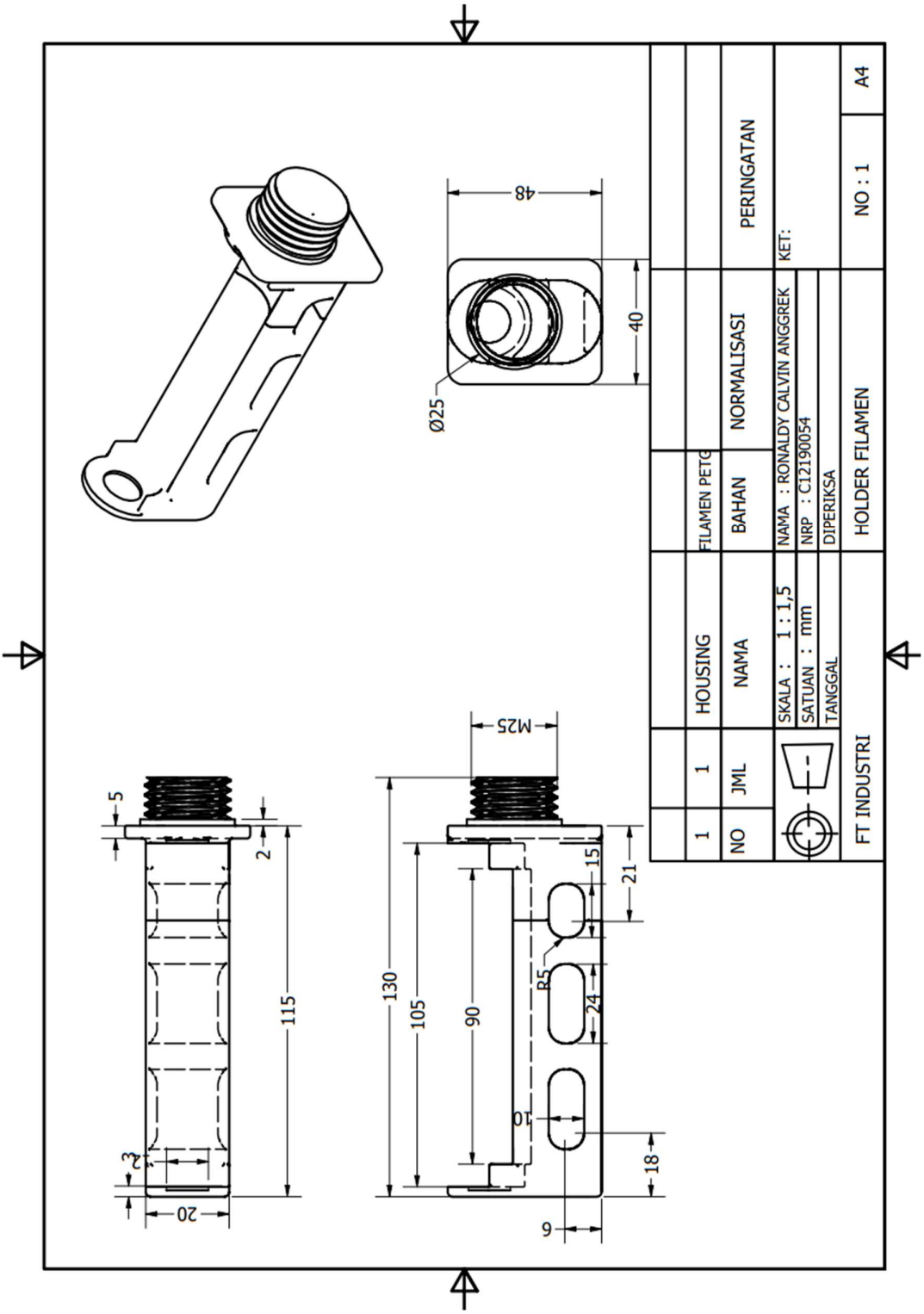
Lampiran 1: Sketch 2D Part Direct Extruder



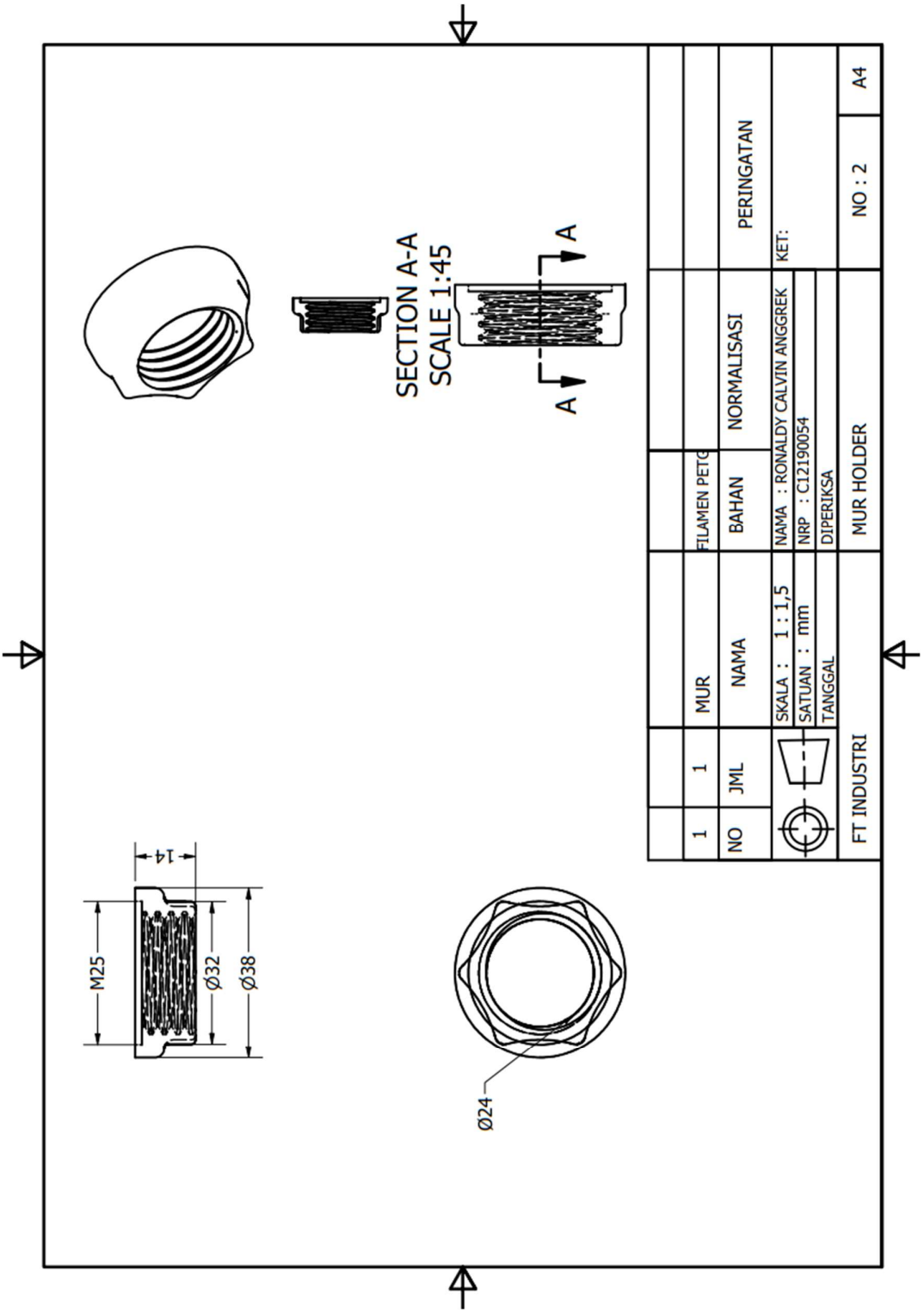
Lampiran 2: Sketch 2D Assembly Direct Extruder



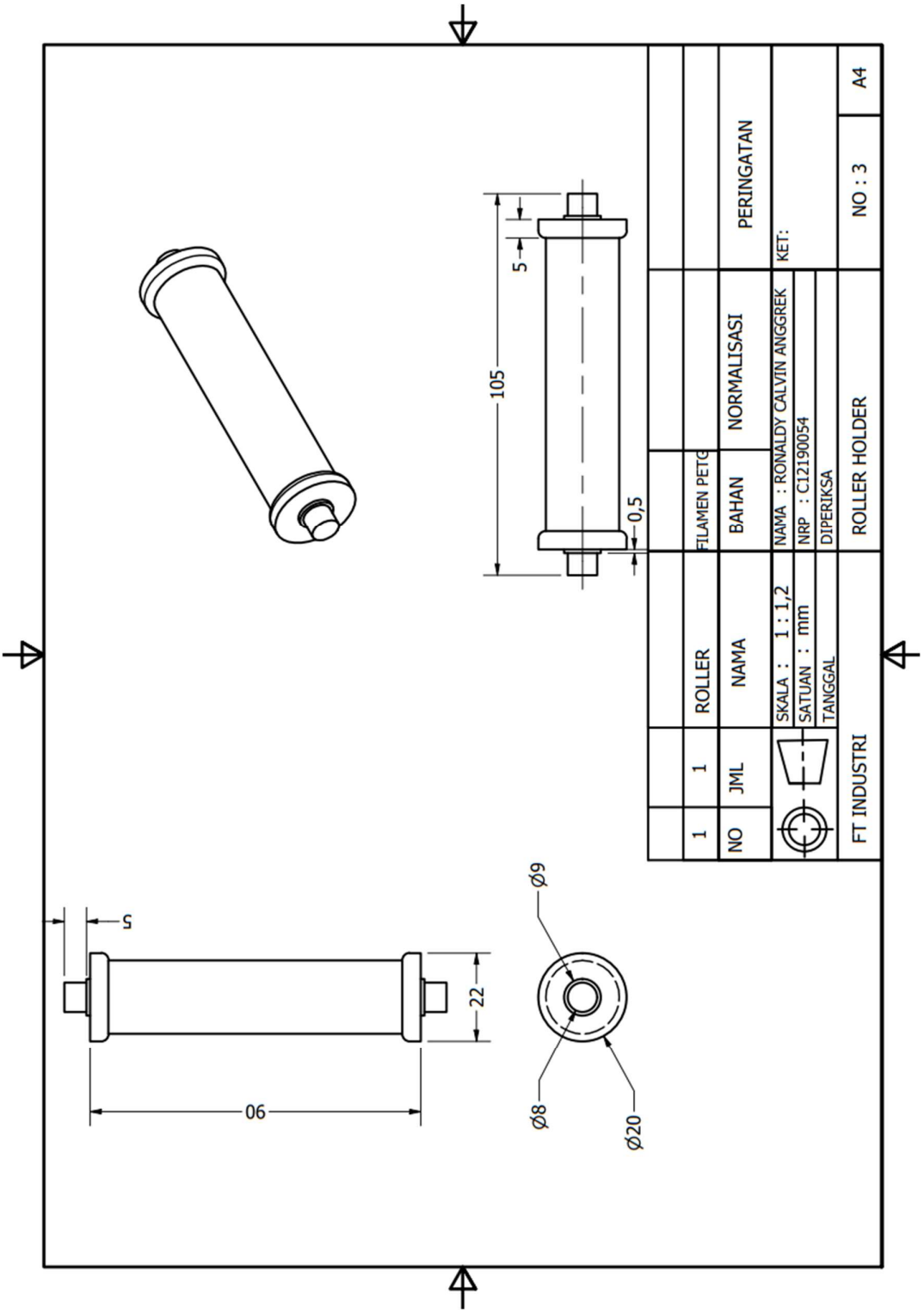
Lampiran 3: Sketch 2D Part Holder Filamen (Housing)



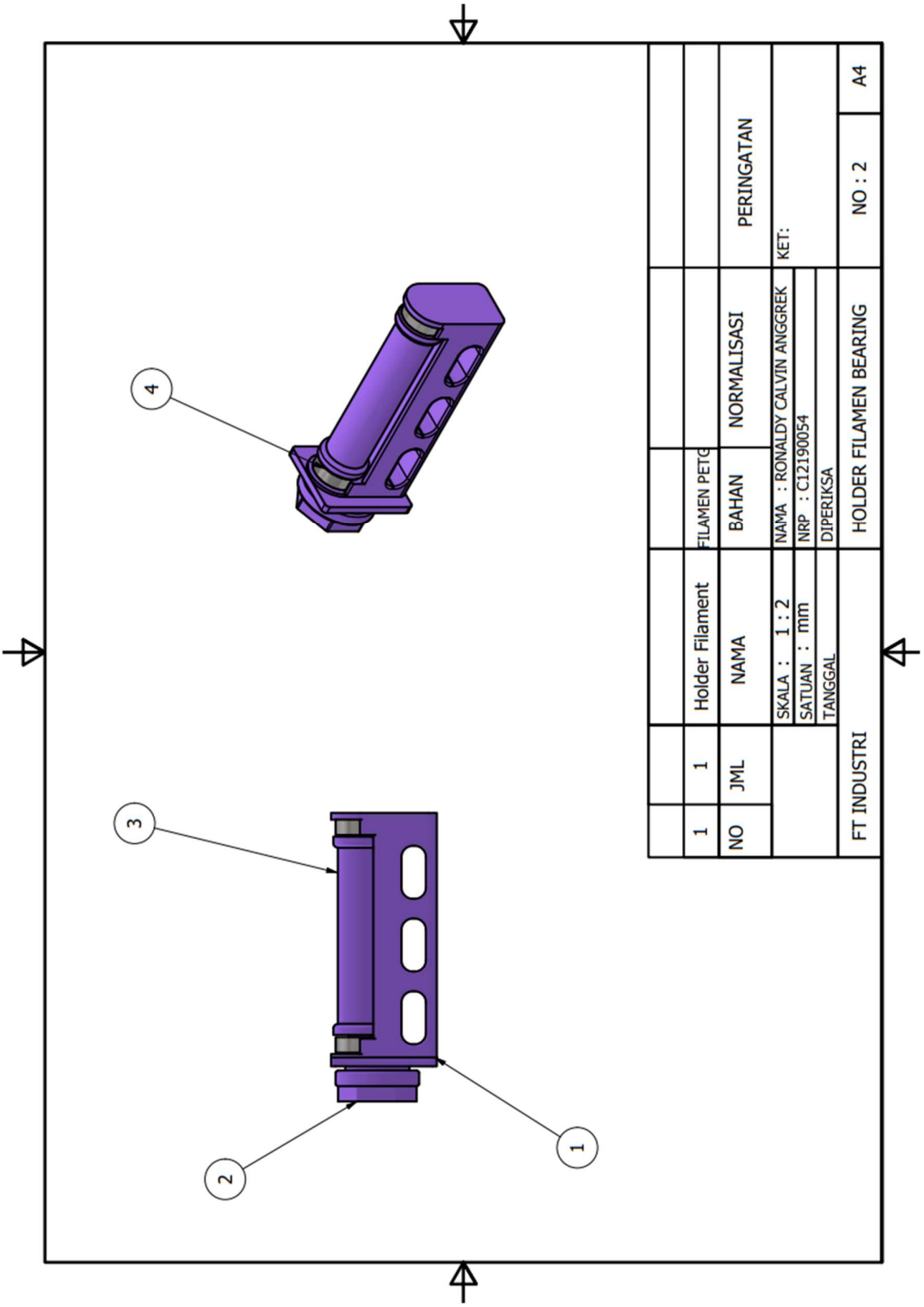
Lampiran 4: Sketch 2D Part Holder Filamen (Mur Holder Filamen)



Lampiran 5: Sketch 2D Part Holder Filamen (Roller)



Lampiran 6: Sketch 2D Assembly Holder Filamen



Lampiran 7: Coding untuk Mengkonfigurasi Gerakan X Y Z Pada 3D Printer Ender 3 Pro

```
# This file contains pin mappings for the stock 2020 Creality Ender 3
# Pro with the 32-bit Creality 4.2.2 board. To use this config, during
# "make menuconfig" select the STM32F103 with a "28KiB bootloader" and
# serial (on USART1 PA10/PA9) communication.

# It should be noted that newer variations of this printer shipping in
# 2022 may have GD32F103 chips installed and not STM32F103. You may
# have to inspect the mainboard to ascertain which one you have. If it
# is the GD32F103 then please select Disable SWD at startup in the
# "make menuconfig" along with the same settings for STM32F103.

# If you prefer a direct serial connection, in "make menuconfig"
# select "Enable extra low-level configuration options" and select
# serial (on USART3 PB11/PB10), which is broken out on the 10 pin IDC
# cable used for the LCD module as follows:
# 3: Tx, 4: Rx, 9: GND, 10: VCC

# Flash this firmware by copying "out/klipper.bin" to a SD card and
# turning on the printer with the card inserted. The firmware
# filename must end in ".bin" and must not match the last filename
# that was flashed.

# See docs/Config_Reference.md for a description of parameters.

[stepper_x]
step_pin: PC2
dir_pin: PB9
enable_pin: !PC3
microsteps: 16
rotation_distance: 40
endstop_pin: ^PA5
```

position_endstop: 0

position_max: 235

homing_speed: 50

[stepper_y]

step_pin: PB8

dir_pin: PB7

enable_pin: !PC3

microsteps: 16

rotation_distance: 40

endstop_pin: ^PA6

position_endstop: 0

position_max: 235

homing_speed: 50

[stepper_z]

step_pin: PB6

dir_pin: !PB5

enable_pin: !PC3

microsteps: 16

rotation_distance: 8

endstop_pin: ^PA7

position_endstop: 0.0

position_max: 250

[extruder]

max_extrude_only_distance: 100.0

step_pin: PB4

dir_pin: PB3

enable_pin: !PC3

microsteps: 16

rotation_distance: 34.406

nozzle_diameter: 0.400

filament_diameter: 1.750

heater_pin: PA1

sensor_type: EPCOS 100K B57560G104F

sensor_pin: PC5

control: pid

tuned for stock hardware with 200 degree Celsius target

pid_Kp: 21.527

pid_Ki: 1.063

pid_Kd: 108.982

min_temp: 0

max_temp: 250

[heater_bed]

heater_pin: PA2

sensor_type: EPCOS 100K B57560G104F

sensor_pin: PC4

control: pid

tuned for stock hardware with 50 degree Celsius target

pid_Kp: 54.027

pid_Ki: 0.770

pid_Kd: 948.182

min_temp: 0

max_temp: 130

[fan]

pin: PA0

[mcu]

serial: /dev/serial/by-path/pci-0000:00:1a.0-usbv2-0:1.4:1.0-port0

restart_method: command

[printer]

kinematics: cartesian

```
max_velocity: 300
max_accel: 3000
max_z_velocity: 5
max_z_accel: 100

# Pin mappings for BL_T port
#[bltouch]
#sensor_pin: ^PB1
#control_pin: PB0

[display]
lcd_type: st7920
cs_pin: PB12
sclk_pin: PB13
sid_pin: PB15
encoder_pins: ^PB14, ^PB10
click_pin: ^!PB2

[virtual_sdcard]
path: /home/cadelinux/printer_4_data/gcodes/
on_error_gcode: CANCEL_PRINT

[pause_resume]
#recover_velocity: 50.
#   When capture/restore is enabled, the speed at which to return to
#   the captured position (in mm/s). Default is 50.0 mm/s.

[gcode_macro CANCEL_PRINT]
description: Cancel the actual running print
rename_existing: CANCEL_PRINT_BASE
gcode:
  ##### get user parameters or use default #####
  {% set client = printer['gcode_macro _CLIENT_VARIABLE']|default({}) %}
```

```
  {% set allow_park = client.park_at_cancel|default(false)|lower == 'true' %}
  {% set retract = client.cancel_retract|default(5.0)|abs %}
  ##### define park position #####
  {% set park_x = "" if (client.park_at_cancel_x|default(none) is none)
        else "X=" ~ client.park_at_cancel_x %}
  {% set park_y = "" if (client.park_at_cancel_y|default(none) is none)
        else "Y=" ~ client.park_at_cancel_y %}
  {% set custom_park = park_x|length > 0 or park_y|length > 0 %}
  ##### end of definitions #####
  # restore idle_timeout time if needed
  {% if printer['gcode_macro RESUME'].restore_idle_timeout > 0 %}
   SET_IDLE_TIMEOUT TIMEOUT={printer['gcode_macro RESUME'].restore_idle_timeout}
  {% endif %}
  {% if (custom_park or not printer.pause_resume.is_paused) and allow_park %}
_TOOLHEAD_PARK_PAUSE_CANCEL {park_x} {park_y} {% endif %}
  _CLIENT_RETRACT LENGTH={retract}
  TURN_OFF_HEATERS
  M106 S0
  {client.user_cancel_macro|default("")}
  SET_GCODE_VARIABLE MACRO=RESUME VARIABLE=idle_state VALUE=False
  # clear pause_next_layer and pause_at_layer as preparation for next print
  SET_PAUSE_NEXT_LAYER ENABLE=0
  SET_PAUSE_AT_LAYER ENABLE=0 LAYER=0
  CANCEL_PRINT_BASE

[gcode_macro PAUSE]
description: Pause the actual running print
rename_existing: PAUSE_BASE
gcode:
  ##### get user parameters or use default #####
  {% set client = printer['gcode_macro _CLIENT_VARIABLE']|default({}) %}
  {% set idle_timeout = client.idle_timeout|default(0) %}
  {% set temp = printer[printer.toolhead.extruder].target if printer.toolhead.extruder != '' else 0
```

```
%}
 {% set restore = False if printer.toolhead.extruder == ''
          else True  if params.RESTORE|default(1)|int == 1 else False %}
 ##### end of definitions #####
 SET_GCODE_VARIABLE MACRO=RESUME VARIABLE=last_extruder_temp VALUE="{{'restore':
restore, 'temp': temp}}"
 # set a new idle_timeout value
 {% if idle_timeout > 0 %}
  SET_GCODE_VARIABLE MACRO=RESUME VARIABLE=restore_idle_timeout
VALUE={printer.configfile.settings.idle_timeout.timeout}
  SET_IDLE_TIMEOUT TIMEOUT={idle_timeout}
 {% endif %}
 PAUSE_BASE
 {client.user_pause_macro|default("")}
 _TOOLHEAD_PARK_PAUSE_CANCEL {rawparams}

[gcode_macro RESUME]
description: Resume the actual running print
rename_existing: RESUME_BASE
variable_last_extruder_temp: {'restore': False, 'temp': 0}
variable_restore_idle_timeout: 0
variable_idle_state: False
gcode:
 ##### get user parameters or use default #####
 {% set client = printer['gcode_macro _CLIENT_VARIABLE']|default({}) %}
 {% set velocity = printer.configfile.settings.pause_resume.recover_velocity %}
 {% set sp_move = client.speed_move|default(velocity) %}
 {% set runout_resume = True if client.runout_sensor|default("") == ""     # no runout
            else True if not printer[client.runout_sensor].enabled  # sensor is disabled
            else printer[client.runout_sensor].filament_detected %} # sensor status
 {% set can_extrude = True if printer.toolhead.extruder == ''          # no extruder defined in
config
           else printer[printer.toolhead.extruder].can_extrude %} # status of active extruder
```

```
 {% set do_resume = False %}
 {% set prompt_txt = [] %}
 ##### end of definitions #####
 #### Printer comming from timeout idle state ####
 {% if printer.idle_timeout.state|upper == "IDLE" or idle_state %}
  SET_GCODE_VARIABLE MACRO=RESUME VARIABLE=idle_state VALUE=False
  {% if last_extruder_temp.restore %}
   # we need to use the unicode (\u00B0) for the ° as py2 env's would throw an error
otherwise
   RESPOND TYPE=echo MSG='{"Restoring \"%s\" temperature to %3.1f\u00B0C, this may
take some time" % (printer.toolhead.extruder, last_extruder_temp.temp) }'
   M109 S{last_extruder_temp.temp}
   {% set do_resume = True %}
  {% elif can_extrude %}
   {% set do_resume = True %}
  {% else %}
   RESPOND TYPE=error MSG='{"Resume aborted !!! \"%s\" not hot enough, please heat up
again and press RESUME" % printer.toolhead.extruder}'
   {% set _d = prompt_txt.append("\"%s\" not hot enough, please heat up again and press
RESUME" % printer.toolhead.extruder) %}
  {% endif %}
 #### Printer comming out of regular PAUSE state ####
 {% elif can_extrude %}
  {% set do_resume = True %}
 {% else %}
  RESPOND TYPE=error MSG='{"Resume aborted !!! \"%s\" not hot enough, please heat up
again and press RESUME" % printer.toolhead.extruder}'
  {% set _d = prompt_txt.append("\"%s\" not hot enough, please heat up again and press
RESUME" % printer.toolhead.extruder) %}
 {% endif %}
 {% if runout_resume %}
  {% if do_resume %}
   {% if restore_idle_timeout > 0 %} SET_IDLE_TIMEOUT TIMEOUT={restore_idle_timeout} {%
```

```
endif %} # restore idle_timeout time
    {client.user_resume_macro|default("")}
    _CLIENT_EXTRUDE
    RESUME_BASE VELOCITY={params.VELOCITY|default(sp_move)}
  {% endif %}
 {% else %}
  RESPOND TYPE=error MSG='{"Resume aborted !!! \"%s\" detects no filament, please load
filament and press RESUME" % (client.runout_sensor.split(" "))[1]}'
  {% set _d = prompt_txt.append("\"%s\" detects no filament, please load filament and press
RESUME" % (client.runout_sensor.split(" "))[1]) %}
 {% endif %}
 ##### Generate User Information box in case of abort #####
 {% if not (runout_resume and do_resume) %}
  RESPOND TYPE=command MSG="action:prompt_begin RESUME aborted !!!"
  {% for element in prompt_txt %}
   RESPOND TYPE=command MSG='{"action:prompt_text %s" % element}'
  {% endfor %}
  RESPOND TYPE=command MSG="action:prompt_footer_button Ok|RESPOND
TYPE=command MSG=action:prompt_end|info"
  RESPOND TYPE=command MSG="action:prompt_show"
 {% endif %}
```