DODATEK A: PROGRAM MAIN.PY - ZARZĄDZAJĄCY

```python
from PySide6.QtCore import *
from PySide6.QtGui import *
from PySide6.QtUiTools import *
from PySide6.QtWidgets import *
import sys, os, time
import utils.postition_calculating as postition_calculating
import utils.communication as comm

#Main window class#
class WindowApp:
    ##Qt Init##
    app = QApplication([])
    ui_file = QFile(os.getcwd() + '/src/utils/form.ui')
    ui_file.open(QFile.ReadOnly)
    loader = QUiLoader()
    window = loader.load(ui_file)
    ##################

    ##Qt variables##
    button_exit = window.button_exit
    button_camera = window.button_camera
    button_add_to_path = window.button_add_to_path
    button_clear_path = window.button_clear_path
    button_cycle_once = window.button_cycle_once
    button_cycle_loop = window.button_cycle_loop
    button_close_gripper = window.button_close_gripper
    button_open_gripper = window.button_open_gripper
    button_refresh = window.button_refresh
    button_set_pos = window.button_set_pos
    button_set_angle = window.button_set_angle
    label_positions = window.label_positions
    label_path = window.label_path
    combo_items = window.combo_items
    lineEdit_x = window.lineEdit_x
    lineEdit_y = window.lineEdit_y
    lineEdit_z = window.lineEdit_z
    lineEdit_fi1 = window.lineEdit_fi1
    lineEdit_fi2 = window.lineEdit_fi2
    ##################

    ##Variables##
    current_path = list()

    def __init__(self):
        ##Adding actions to buttons##
        self.button_exit.clicked.connect(sys.exit)
        self.button_camera.clicked.connect(postition_calculating.loop)
        self.button_add_to_path.clicked.connect(self.add_to_path)
        self.button_clear_path.clicked.connect(self.clear_path)
        self.button_cycle_once.clicked.connect(self.cycle_once)
        self.button_cycle_loop.clicked.connect(self.cycle_loop)
        self.button_refresh.clicked.connect(self.refresh)
        self.button_set_pos.clicked.connect(self.set_position)
        self.button_set_angle.clicked.connect(self.set_angle)
        self.button_open_gripper.clicked.connect(postition_calculating.open_gripper)
        self.button_close_gripper.clicked.connect(postition_calculating.close_gripper)
        ############################

        ##Adding items to postions list##
        self.add_items_to_label()

        ##Append list to comboBox##
        self.add_items_to_combo()

        ##Show window##
        self.show_window()

```

```
68      def set_position(self):
69          given_pos_X = float(self.lineEdit_x.text())
70          given_pos_Y = float(self.lineEdit_y.text())
71          given_pos_Z = float(self.lineEdit_z.text())
72
73          fis = postition_calculating.inverse_kinematics(given_pos_X,given_pos_Y)
74
75          comm.send_fi_to_Arduino(given_pos_Z, 1)
76          comm.send_fi_to_Arduino(fis[0], 2)
77          comm.send_fi_to_Arduino(fis[1], 3)
78
79      def set_angle(self):
80          givem_fi1 = float(self.lineEdit_fi1.text())
81          givem_fi2 = float(self.lineEdit_fi2.text())
82
83          comm.send_fi_to_Arduino(givem_fi1, 2)
84          comm.send_fi_to_Arduino(givem_fi2, 3)
85
86      def refresh(self):
87          self.update_path_label()
88          self.add_items_to_label()
89          self.add_items_to_combo()
90
91      def cycle_once(self):
92          for positions in self.current_path_positions:
93              comm.send_fi_to_Arduino(positions[4],1)
94              comm.send_fi_to_Arduino(positions[0],2)
95              comm.send_fi_to_Arduino(positions[1],3)
96              time.sleep(2)
97
98      def cycle_loop(self):
99          pass
100
101     def show_window(self):
102         self.window.show()
103
104     #After clicking button, func adds selected position to route#
105     def add_to_path(self):
106         curr_position = self.combo_items.currentText()
107         self.current_path.append(curr_position)
108         self.update_path_label()
109
110     #Clears whole saved route#
111     def clear_path(self):
112         self.current_path = list()
113         self.current_path_positions = list()
114         self.update_path_label()
115
116     #Get list of num of positions eg. ['1', '2', ...] for comboBox#
117     def get_list_of_pos_numbers(self):
118         data_to_return = list()
119         for index in range(len(postition_calculating.get_positions()) // 5):
120             data_to_return.append(str(index + 1))
121         return data_to_return
122
123     #Append to comboBox#
124     def add_items_to_combo(self):
125         self.combo_items.addItems(list())
126         self.combo_items.addItems(self.get_list_of_pos_numbers())
127
128     #Updates label with current route#
129     def update_path_label(self):
130         positions = postition_calculating.get_positions()
131         self.current_path_positions = list()
132         oneline_text = ""
133         text_to_post = ""
134
135         for index, value in enumerate(self.current_path):
136             value = int(value)
137             curr_pos = positions[(value - 1) * 5:(value - 1) * 5 + 5]
138             self.current_path_positions.append(curr_pos)
139             for i, val in enumerate(curr_pos):
140                 if i == 0:
```

```
                    oneline_text += f'{index + 1}. '
                elif i == 1:
                    pass
                elif i == 2:
                    oneline_text += f'X:{val} '
                elif i == 3:
                    oneline_text += f'Y:{val} '
                elif i == 4:
                    oneline_text += f'Z:{val}\n'
                    text_to_post = oneline_text
        self.label_path.setText(text_to_post)

    #Add postions from positions.txt to list_positions#
    def add_items_to_label(self):
        positions = postition_calculating.get_positions()
        text_to_post = ""
        oneline_text = ""
        count = 0
        for index, value in enumerate(positions):
            if count == 0:
                oneline_text = f'{index // 5 + 1}.'
            elif count == 1:
                pass
            elif count == 2:
                oneline_text += f' X:{value}'
            elif count == 3:
                oneline_text += f' Y:{value}'
            elif count == 4:
                oneline_text += f' Z:{value}\n'
                text_to_post += oneline_text
                oneline_text = ""
                count = 0
                continue
            count += 1
        self.label_positions.setText(text_to_post)

if __name__ == "__main__":
    window_application = WindowApp()
    sys.exit(window_application.app.exec())
```

DODATEK B: PROGRAM POSITION_CALCULATING.PY - OBLICZENIOWY

```python
from cvzone.HandTrackingModule import HandDetector
import cvzone
import cv2
import numpy as np
import math
import os
import utils.communication as comm

##Camera init##
camera = cv2.VideoCapture(0)
if not camera.isOpened():
    camera.open("http://192.168.0.20:8080")
if not camera.isOpened():
    camera.open("http://192.168.0.25:8080")
if not camera.isOpened():
    camera.open("http://192.168.43.162:8080")
if not camera.isOpened():
    print("Please connect camera")

##Get camera variables##
camera_width = camera.get(cv2.CAP_PROP_FRAME_WIDTH)
camera_height = camera.get(cv2.CAP_PROP_FRAME_HEIGHT)
camera_suspension_height = 100

##Variables##
robot_max_range_CM = [17,17,15.5] # [X, Y, Z]
robot_arm_lengths = [11.37,6.5,10] # [A1, A2, A3]
is_gesture_grip = False
is_gesture_position = False
positions_file = os.getcwd() + "/src/utils/positions.txt"
last_pos = (0,0,0,0,0)
counted_frame_to_send = 0
how_many_messages_send = 20
pos_round = 2
is_gripper_closed = False
stepper_motor_max_step = 4000

##Distance shenanigans##
x = [300, 245, 200, 170, 145, 130, 112, 103, 93, 87, 80, 75, 70, 67, 62, 59, 57]
y = [20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
coff = np.polyfit(x, y, 2)  # y = Ax^2 + Bx + C
detector = HandDetector(detectionCon=0.8, maxHands=1)

def loop():
    while True:
        img = get_frame()
        try:
            if img == False:
                cv2.destroyAllWindows()
                return
        except:
            ##Print image##
            cv2.imshow("Image", img)
            cv2.waitKey(1)
            ################

#Save current position to file#
def save_postion(fi1,fi2,X,Y,Z):
    if X == 0 and Y == 0 and Z == 0:
        return
    pos = [str(fi1) + '\n', str(fi2) + '\n',str(X) + '\n', str(Y) + '\n', str(Z) + '\n']
    print(pos)
    with open(positions_file, "a") as file:
        file.writelines(pos)

#Get postitions from file#
def get_positions():
```

```
    with open(positions_file, "r") as file:
        data = list()
        for line in file:
            data.append(float(line))
    return data

#Returns given position#
def get_position(pos):
    with open(positions_file, "r") as file:
        data = list()
        for index, line in enumerate(file):
            if index > (pos - 1) * 5 - 1:
                data.append(float(line))
                if index >= (pos - 1) * 5 + 5:
                    break
    return data

#One frame#
def get_frame():
    global is_gesture_grip, is_gesture_position, last_pos, counted_frame_to_send,
    how_many_messages_send, is_gripper_closed
    _, img = camera.read()
    hands = detector.findHands(img, draw=False)

    if hands:
        ##Reading from mediapipe output##
        hand1 = hands[0]
        lmList = hand1["lmList"]
        lmList = hands[0]['lmList']
        x, y, _, _ = hands[0]['bbox']
        x1, y1, _ = lmList[5]
        x2, y2, _ = lmList[17]
        ######################

        ##Measuring distance##
        distance = int(math.sqrt((y2 - y1) ** 2 + (x2 - x1) ** 2))
        A, B, C = coff
        distanceCM = A * distance ** 2 + B * distance + C
        ######################

        ##Gestures recognition##
        fingers = detector.fingersUp(hands[0])
        if fingers == [1, 1, 1, 1, 1]:
            cvzone.putTextRect(img, f'{int(distanceCM)} cm X:{int(lmList[8][0])} Y:{int(
    camera_height - lmList[8][1])}', (x+5, y-10), border=2)
            is_gesture_grip = False
            is_gesture_position = False
        elif fingers == [1,1,0,1,1]:
            cvzone.putTextRect(img, f'{int(distanceCM)} cm X:{int(lmList[8][0])} Y:{int(
    camera_height - lmList[8][1])}', (x+5, y-10), border=3)
            if not is_gesture_grip:
                is_gesture_grip = True
                if is_gripper_closed:
                    is_gripper_closed = False
                    open_gripper()
                else:
                    is_gripper_closed = True
                    close_gripper()
        elif fingers == [0,1,0,0,0]:
            #cvzone.putTextRect(img, f'{int(distanceCM)} cm X:{int(lmList[8][0])} Y:{int
    (camera_height - lmList[8][1])}', (x+5, y), border=3)
            cv2.circle(img, (lmList[8][0],lmList[8][1]), 10, (0,0,255),10)
            last_pos = calculate_kinematics(lmList, distanceCM)
            if counted_frame_to_send == how_many_messages_send:
                comm.send_fi_to_Arduino(last_pos[4], 1)
                comm.send_fi_to_Arduino(last_pos[0], 2)
                comm.send_fi_to_Arduino(last_pos[1], 3)
                counted_frame_to_send = 0
            else:
                counted_frame_to_send += 1
            is_gesture_position = False
        elif fingers == [0,1,1,0,0]:
            if not is_gesture_position:
```

```
                is_gesture_position = True
                save_postion(last_pos[0],last_pos[1],round(last_pos[2],pos_round),round(last_pos[3],pos_round),round(last_pos[4],pos_round))
            elif fingers == [0,0,0,1,1]:
                return False
            ############################
        return img


def inverse_kinematics(robot_X,robot_Y):
    ##Inverse kinematics##
        try:
            M = (robot_X**2 + robot_Y**2 - robot_arm_lengths[0]**2 - robot_arm_lengths[1]**2)/(2*robot_arm_lengths[0]*robot_arm_lengths[1])
            #fi2 = np.arctan((-np.sqrt(1-M**2))/(M))
            fi2 = np.arccos(M)
            fi1 = np.arctan(robot_Y/robot_X)-np.arctan((robot_arm_lengths[1]*np.sin(fi2))/(robot_arm_lengths[0]+robot_arm_lengths[1] * np.cos(fi2)))
        except:
            print("Division by zero!!")
            return

        ##OUTPUT of inverse kinematics##
        fi1_deg = np.rad2deg(fi1)
        fi2_deg = np.rad2deg(fi2)
        ######################
        return (fi1_deg,fi2_deg)

##Calculating inverse kinematics##
def calculate_kinematics(lmList,distanceCM):
        ##Counting robot X Y Z based on camera output##
        #robot_X = ((lmList[8][0] - camera_width//2)/camera_width) * robot_max_range_CM[0]
        robot_X = ((lmList[8][0])/camera_width) * robot_max_range_CM[0]
        robot_Y = (1 - lmList[8][1]/camera_height) * robot_max_range_CM[1]
        robot_Z = (1 - distanceCM/camera_suspension_height) * robot_max_range_CM[2]

        #Convert cm to steps for stepper motor
        robot_Z = robot_Z / robot_max_range_CM[2] * stepper_motor_max_step
        ####################

        out_kinematics = inverse_kinematics(robot_X, robot_Y)


        fi1_deg = out_kinematics[0]
        fi2_deg = out_kinematics[1]
        print(f'FI1:{fi1_deg} FI2:{fi2_deg} Z:{robot_Z}\n') # Print angles #
        return (fi1_deg,fi2_deg,robot_X, robot_Y, robot_Z)

def close_gripper():
    comm.send_gripper_to_Arduino('C')

def open_gripper():
    comm.send_gripper_to_Arduino('O')

```

DODATEK C: PROGRAM COMMUNICATION.PY - KOMUNIKACYJNY

```
import serial

##Serial Port Init##
ser = serial.Serial()
ser.baudrate = 9600
ser.port = '/dev/ttyACM0'   #DON'T CHANGE
ser.timeout = 1
try:
    ser.open()
except:
    try:
        ser.port = '/dev/ttyACM1'   #CHANGE HERE
        ser.open()
    except:
        try:
            ser.port = 'COM4'   #CHANGE HERE
            ser.open()
        except:
            print("Cannot find Arduino")
            pass


def send_to_Arduino(data_to_send):
    print(data_to_send)
    ser.write(data_to_send)

def send_fi_to_Arduino(data, arm_num):
    try:
        if arm_num == 2:
            data = int(data)
        elif arm_num == 3:
            data = -int(data)
            data = data + 90
        if data < 0 :
            data = 0
        data_to_send = str(arm_num) + str(data) + '\n'
        send_to_Arduino(data_to_send.encode())
    except:
        "Cannot send data to Arduino!!"

def send_gripper_to_Arduino(gripper_state):
    data_to_send = str(gripper_state) + '\n'
    send_to_Arduino(data_to_send.encode())

```

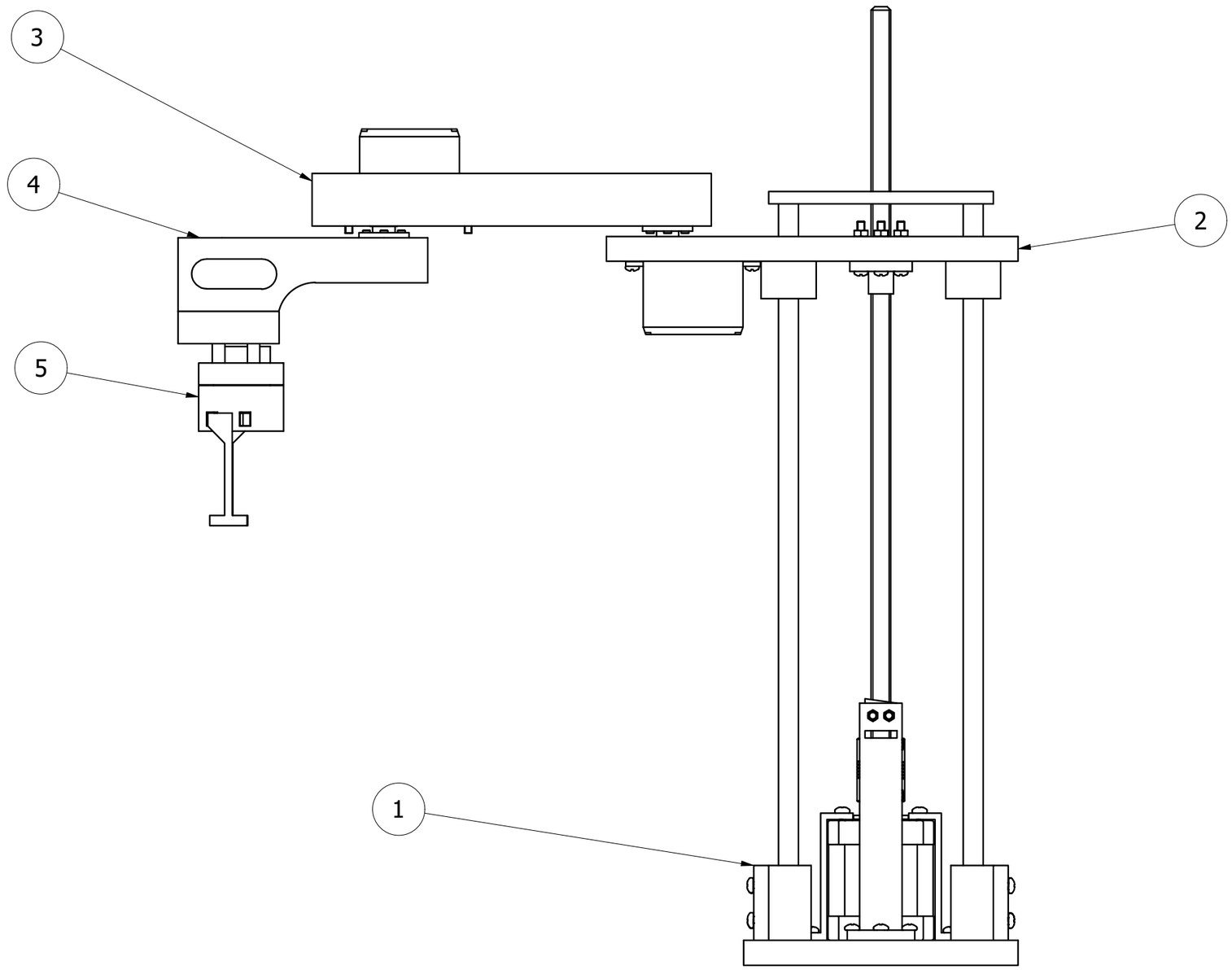
DODATEK D: PROGRAM ARDUINO MAIN.INO

```
#include <Servo.h>
#include <AccelStepper.h>

#define motorInterfaceType 1
#define home_switch 12

//Variables
String incomingData ;
int toIntVal;
const int max_stepper_range = 4000;

//Servos pins
const int servo_fi1_pin = 11;
const int servo_fi2_pin = 5;
const int servo_gripper_pin = 6;

//Stepper pins
const int dirPin = 2;
const int stepPin = 9;

//Servo and stepper init
AccelStepper myStepper(motorInterfaceType, stepPin, dirPin);
Servo servo_fi1, servo_fi2, servo_gripper;

void setup() {
  Serial.begin(9600);

  servo_fi1.attach(servo_fi1_pin);
  servo_fi2.attach(servo_fi2_pin);
  servo_gripper.attach(servo_gripper_pin);

  pinMode(home_switch, INPUT_PULLUP); //Home switch init
  home_stepper();
  set_stepper_parameters();
}
void loop() {
  if (Serial.available() > 0) {
      incomingData = Serial.readStringUntil('\n');
      if(incomingData[0] == '1')//Z-Axis
      {
        incomingData.remove(0,1);
        toIntVal = incomingData.toInt();
        move_stepper_to_possition(toIntVal);
      }
      else if(incomingData[0] == '2')//Fi1
      {
        incomingData.remove(0,1);
        toIntVal = incomingData.toInt();
        servo_fi1.write(toIntVal);
      }
      else if(incomingData[0] == '3')//Fi2
      {
        incomingData.remove(0,1);
        toIntVal = incomingData.toInt();
        servo_fi2.write(toIntVal);
      }
      else if(incomingData[0] == 'O') // Open
      {
          servo_gripper.write(80);
      }
      else if(incomingData[0] == 'C') // Close
      {
          servo_gripper.write(180);
      }
  }
  delay(200);
}
```

```
68
69 void move_stepper_to_possition(int go_to_step)
70 {
71   if(go_to_step >= max_stepper_range)
72   {
73     go_to_step = max_stepper_range;
74   }
75   if(go_to_step <= 0)
76   {
77     go_to_step = 0;
78   }
79
80   myStepper.moveTo(-go_to_step);
81
82   while(myStepper.distanceToGo() != 0)
83   {
84     myStepper.run();
85   }
86
87 }
88
89 void set_stepper_parameters()
90 {
91   myStepper.setMaxSpeed(2000);
92   myStepper.setAcceleration(700);
93   myStepper.setSpeed(1000);
94 }
95
96 void home_stepper()
97 {
98     int curr_step = 0;
99
100    //Set maxSpeed, Acceleration and Speed for homing procedure
101    myStepper.setMaxSpeed(1000);
102    myStepper.setAcceleration(20);
103    myStepper.setSpeed(10);
104
105    //Move down till home switch is pressed
106    while(!digitalRead(home_switch))
107    {
108      delay(2);
109      myStepper.moveTo(curr_step);
110      curr_step++;
111      myStepper.run();
112    }
113
114    //Set current position to 0
115    myStepper.setCurrentPosition(0);
116 }
117
```

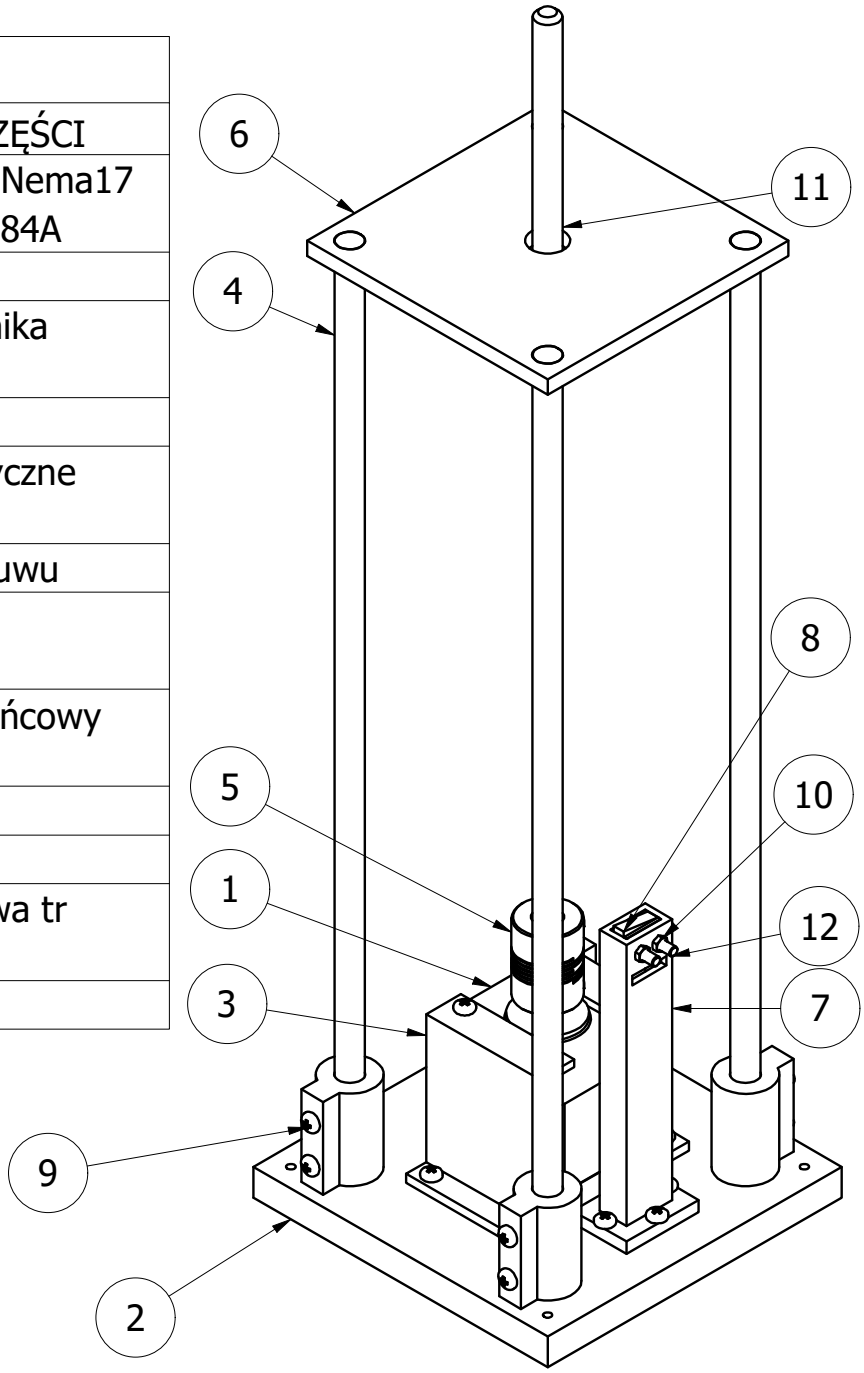
DODATEK E: RYSUNKI TECHNICZNE

LISTA CZĘŚCI		
POZYCJA	ILOŚĆ	NUMER CZĘŚCI
1	1	Podstawa
2	1	Przesów
3	1	Ramie 1
4	1	Ramie 2
5	1	Chwytak

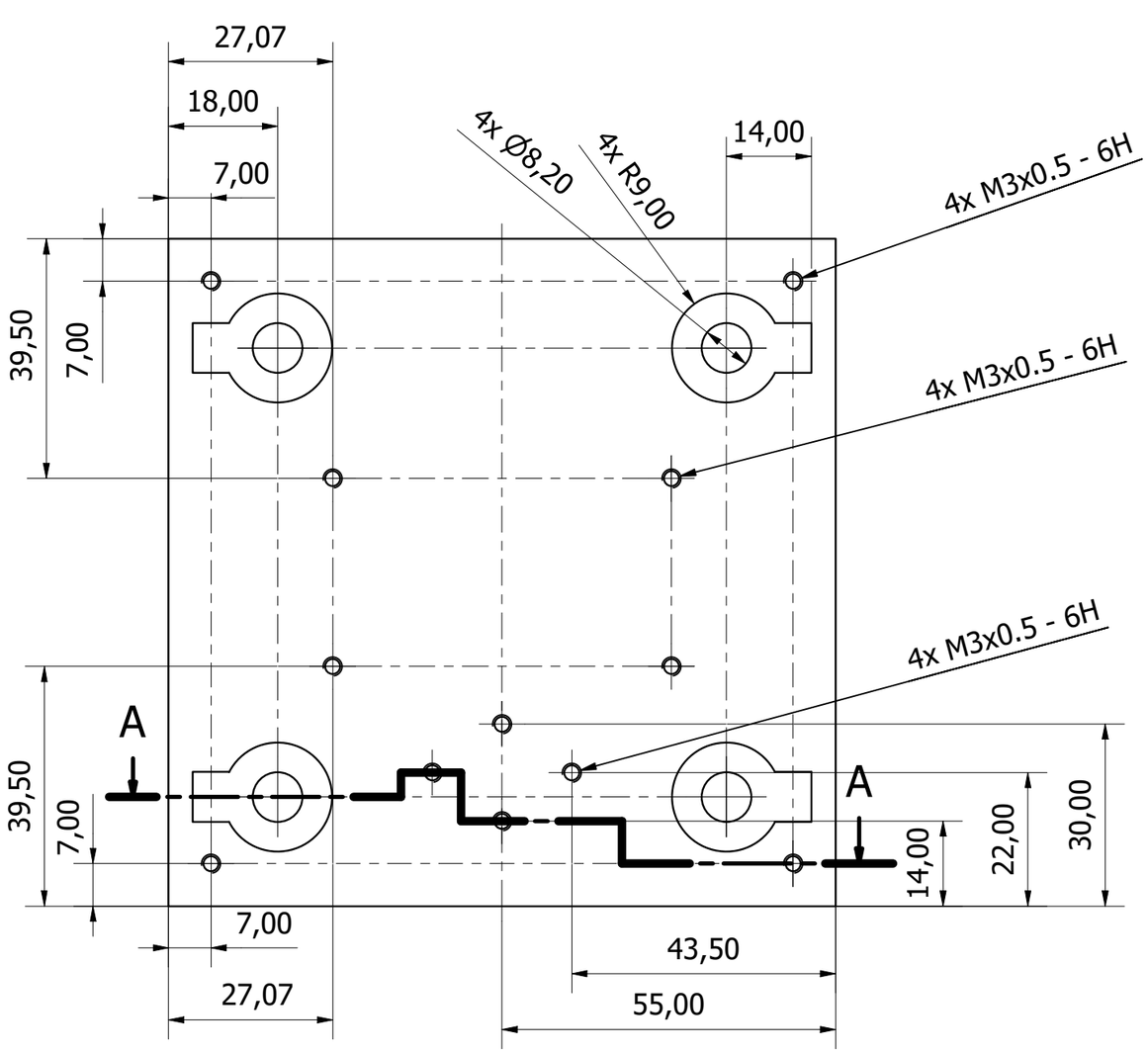


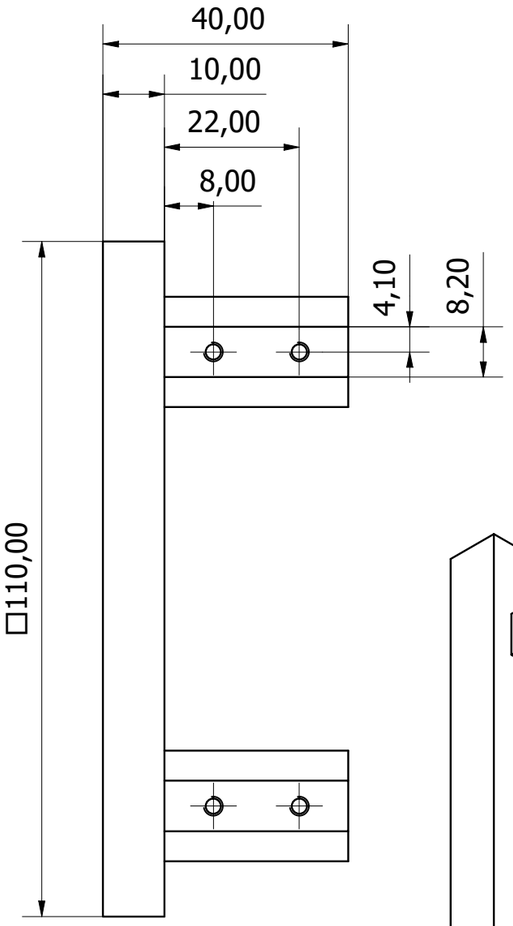
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					06.12.2022	
				Manipulator SCARA		
				Rysunek złożeniowy	Wydanie 1	Arkusz 1 / 1

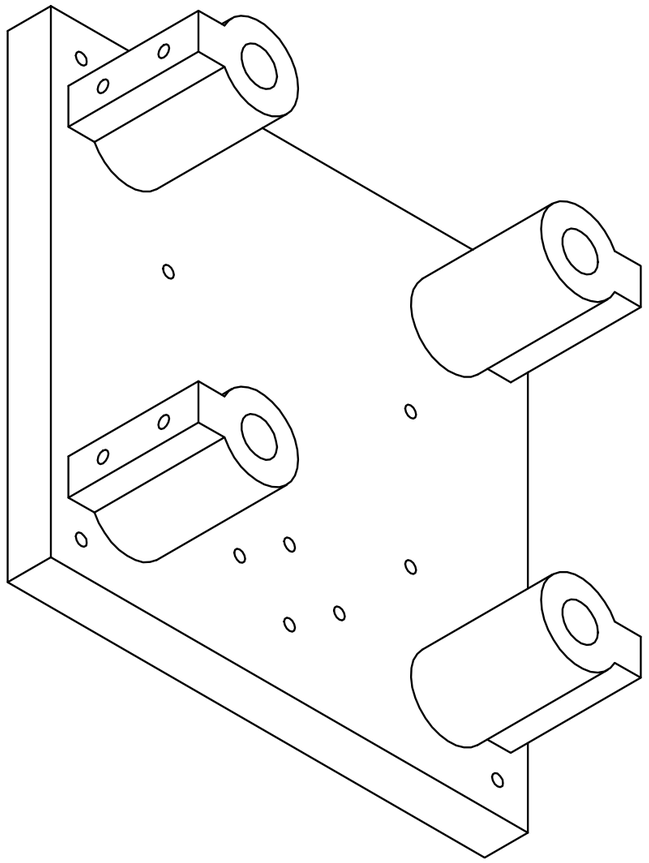
LISTA CZĘŚCI		
POZYCJA	ILOŚĆ	NUMER CZĘŚCI
1	1	Silnik krokowy Nema17 SY42STH47-1684A
2	1	Podstawa
3	2	Mocowanie silnika przesuwu
4	4	Wałek 8 mm
5	1	Sprzęgło elastyczne 5x8mm
6	1	Pokrywa przesuwu
7	1	Stojak czujnika krańcowego
8	1	Przełącznik krańcowy MSW-22
9	20	Śruba M3x12
10	2	Nakrętka M2
11	1	Sruba trapezowa tr 8x1,5
12	2	Śruba M3x16



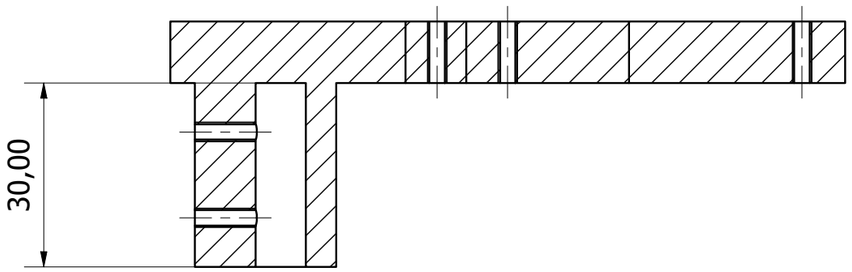
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	
			Podstawa			
			Rysunek złożeniowy		Wydanie 1	Arkusz 1 / 1





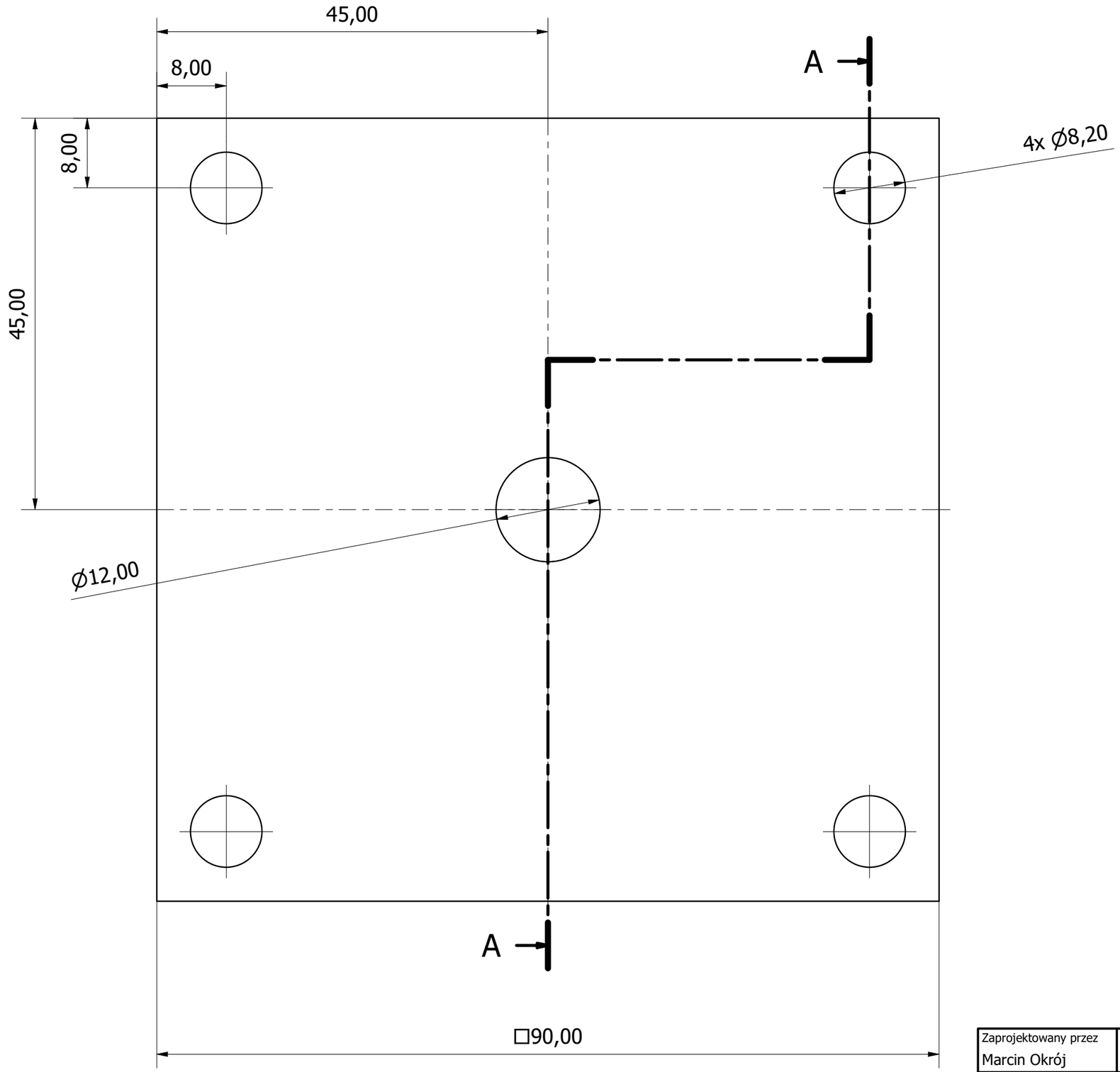


A-A (1 : 1)

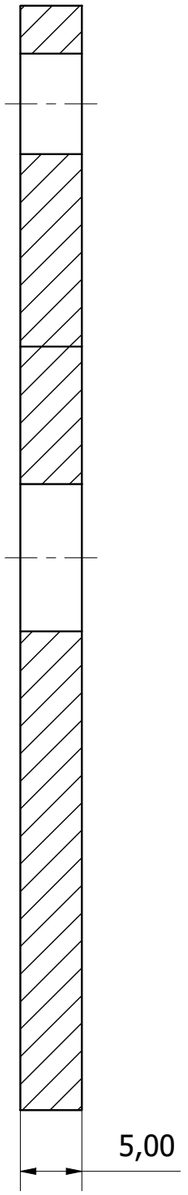


Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					04.12.2022	

	Podstawa		
	Rysunek techniczny	Wydanie 1	Arkusz 1 / 1

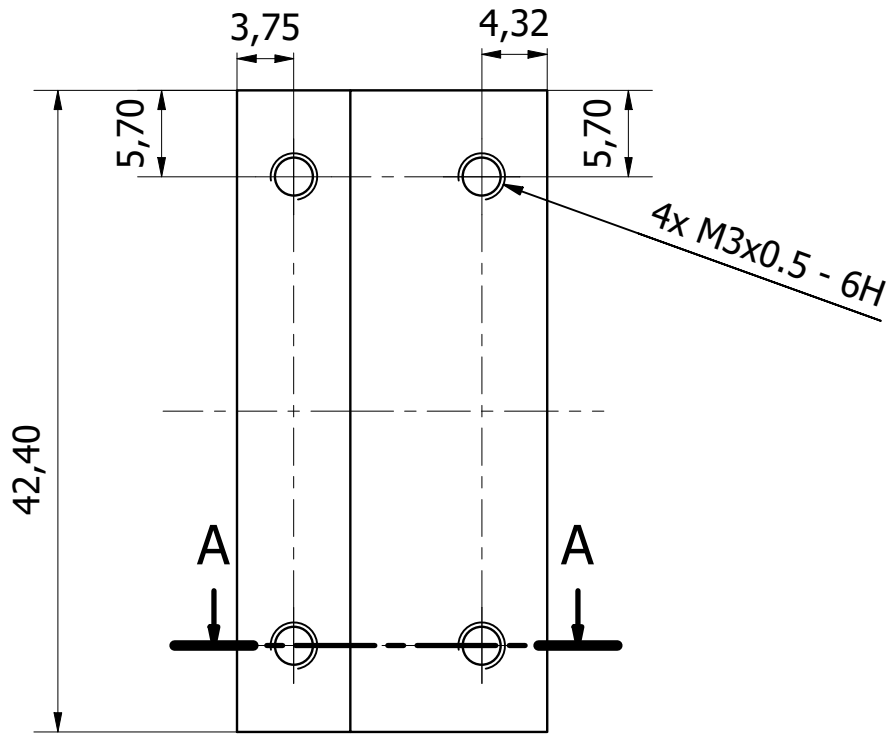


A-A (2 : 1)

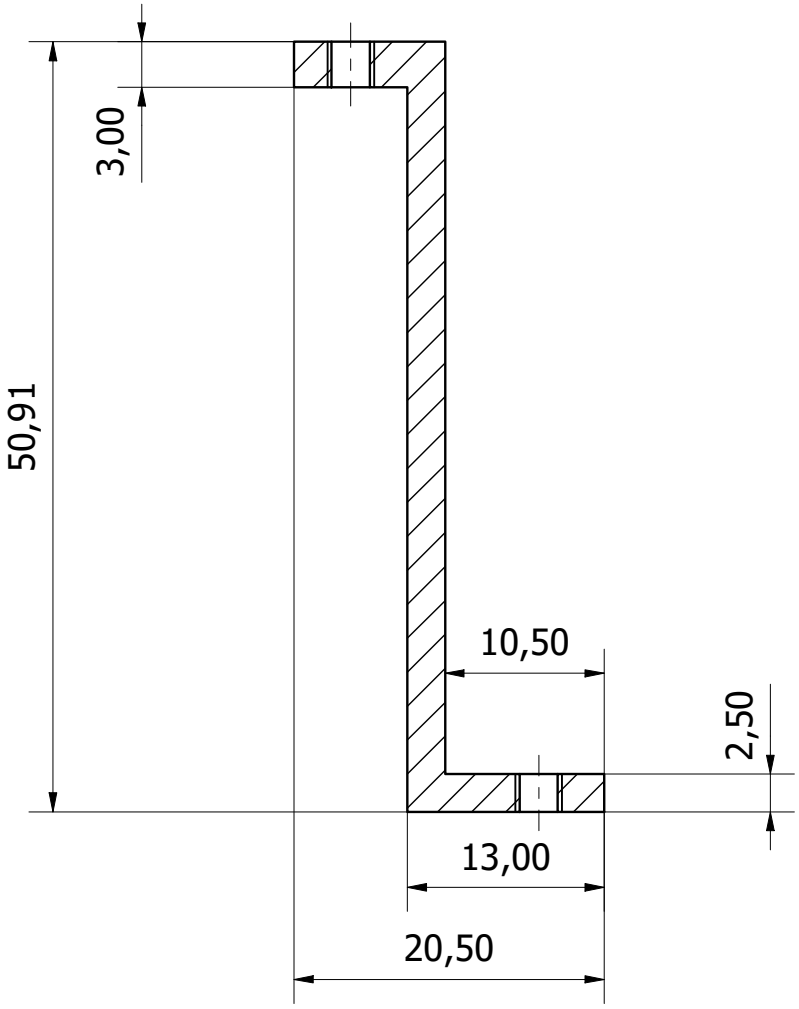


Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					04.12.2022	

	Pokrywa przesuwu		
Rysunek techniczny		Wydanie 1	Arkusz 1 / 1

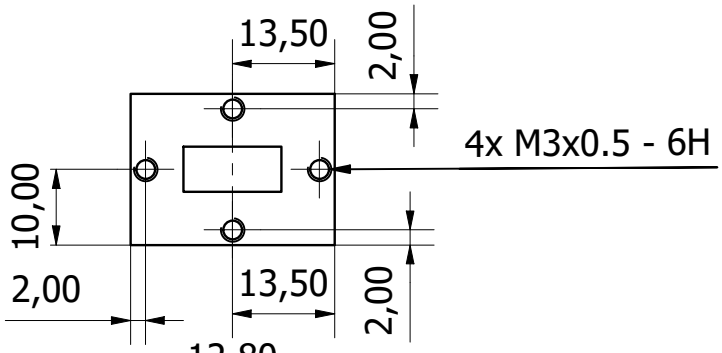


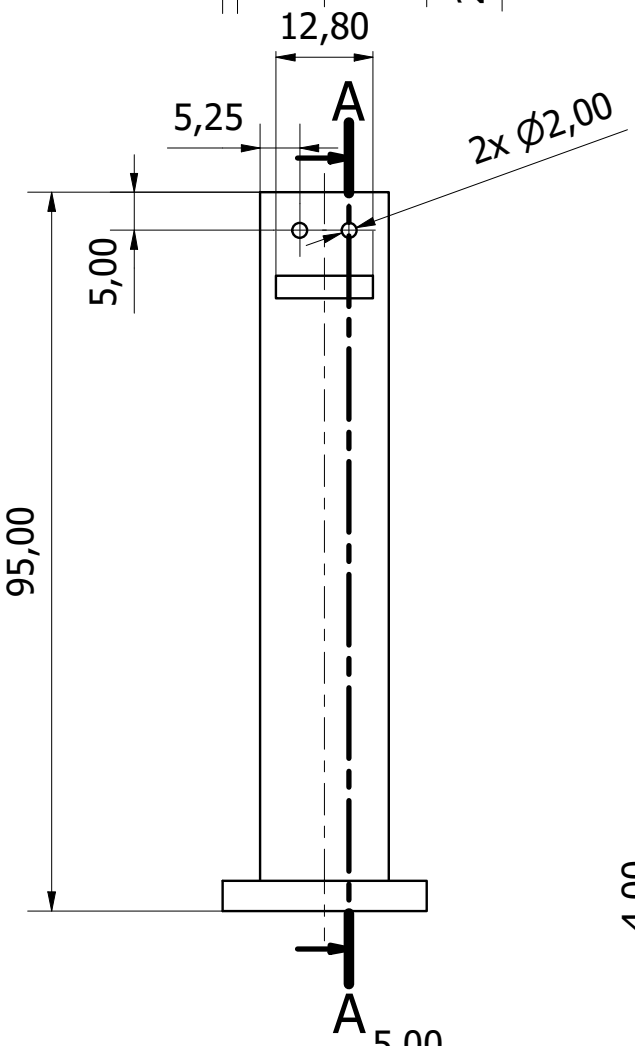
A-A (2 : 1)



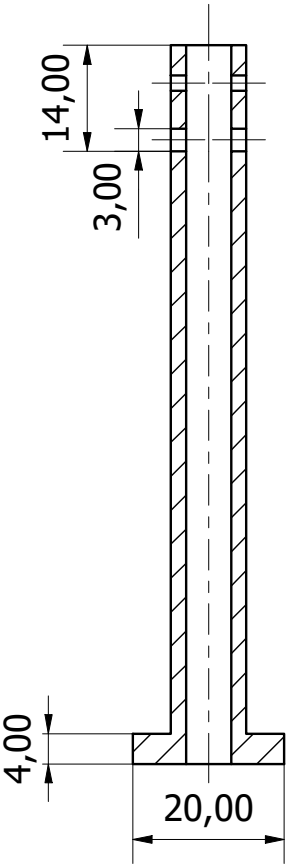
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					04.12.2022	

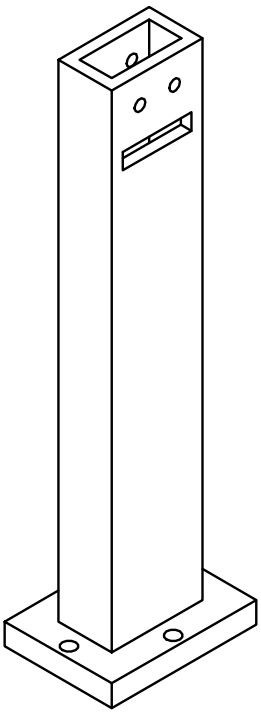
	Mocowanie silnika przesuwu		
	Rysunek techniczny	Wydanie 1	Arkusz 1 / 1

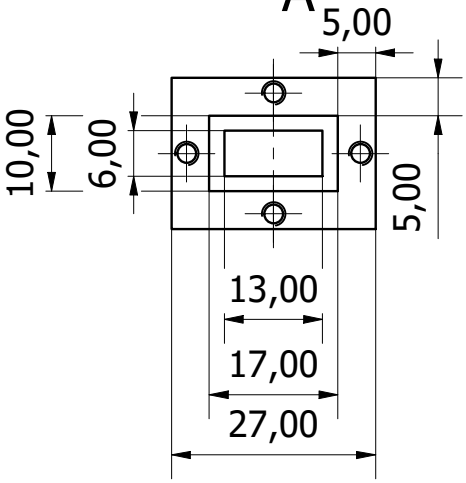




A-A (1 : 1)



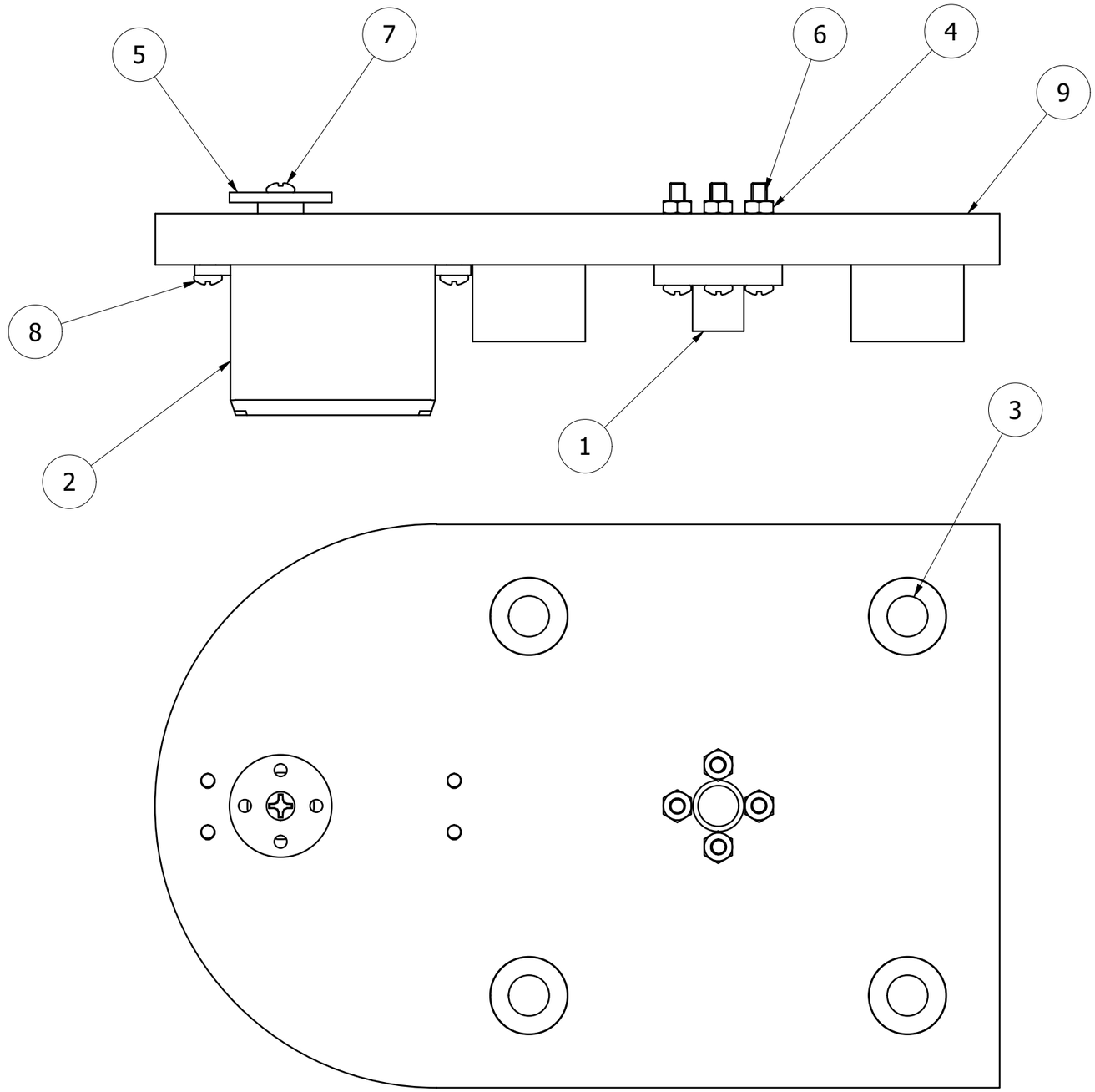




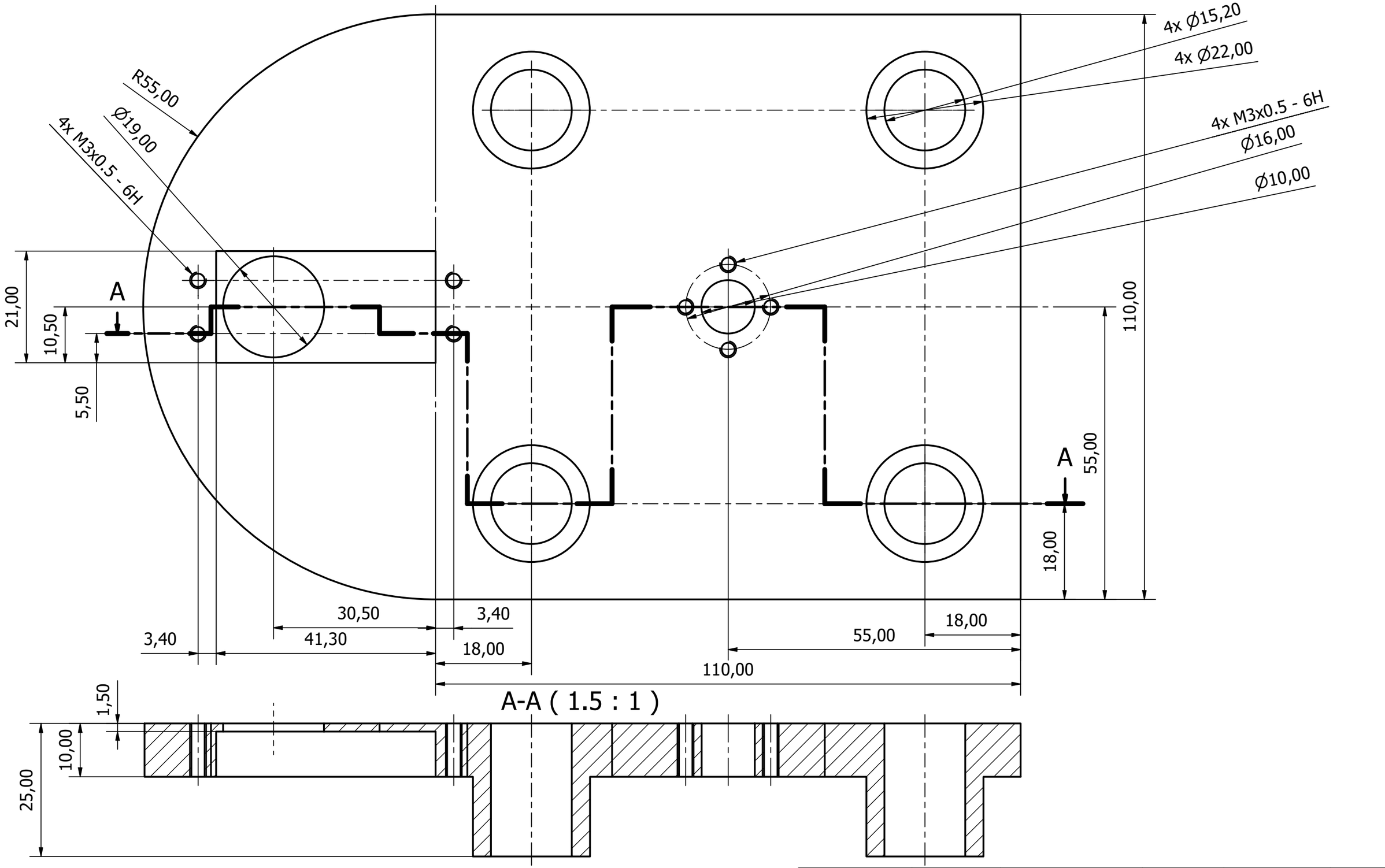
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					04.12.2022	

	Stojak czujnika krańcowego		
	Rysunek techniczny	Wydanie 1	Arkusz 1 / 1

LISTA CZĘŚCI		
POZYCJA	ILOŚĆ	NUMER CZĘŚCI
1	1	Nakrętka śruby trapezowej tr8x1,5
2	1	MG996R Serwo
3	4	Łożysko liniowe LM8UU
4	4	Nakrętka M3
5	1	Orczyk
6	4	Śruba M3x20
7	1	Śruba M3x8
8	4	Śruba M3x12
9	1	Przesuw



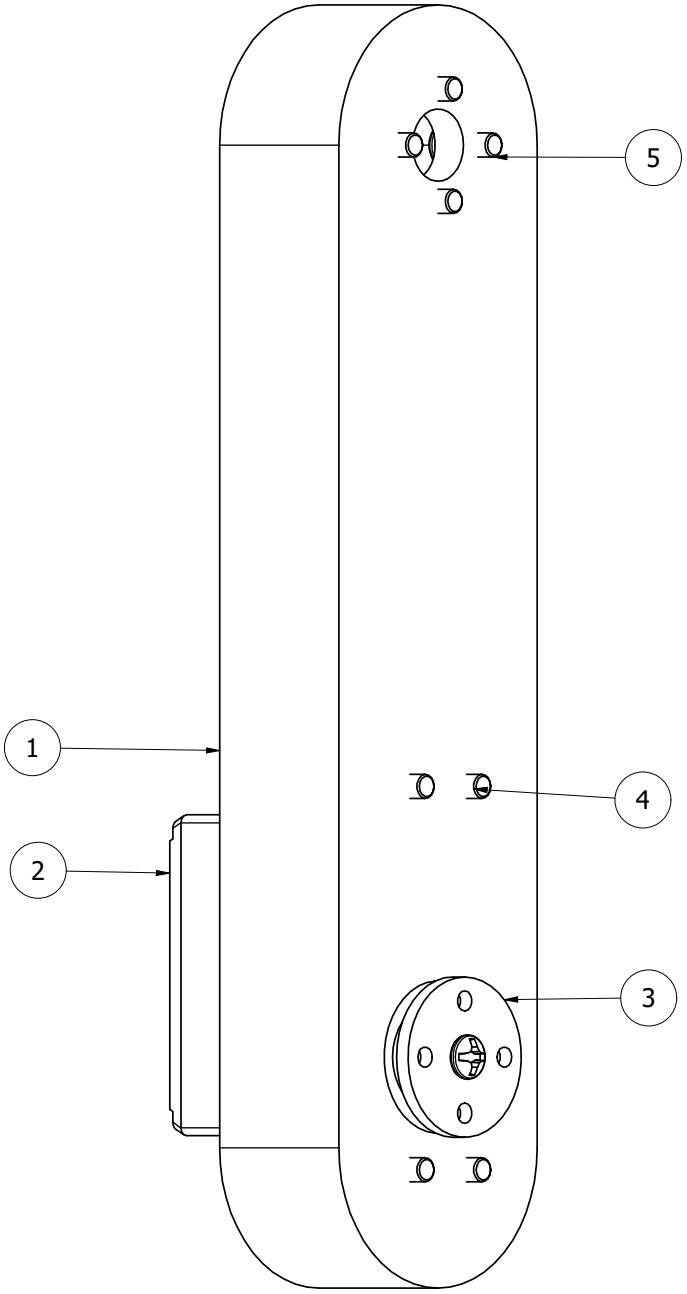
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	
				Przesuw		
				Rysunek złożeniowy	Wydanie 1	Arkusz 1 / 1



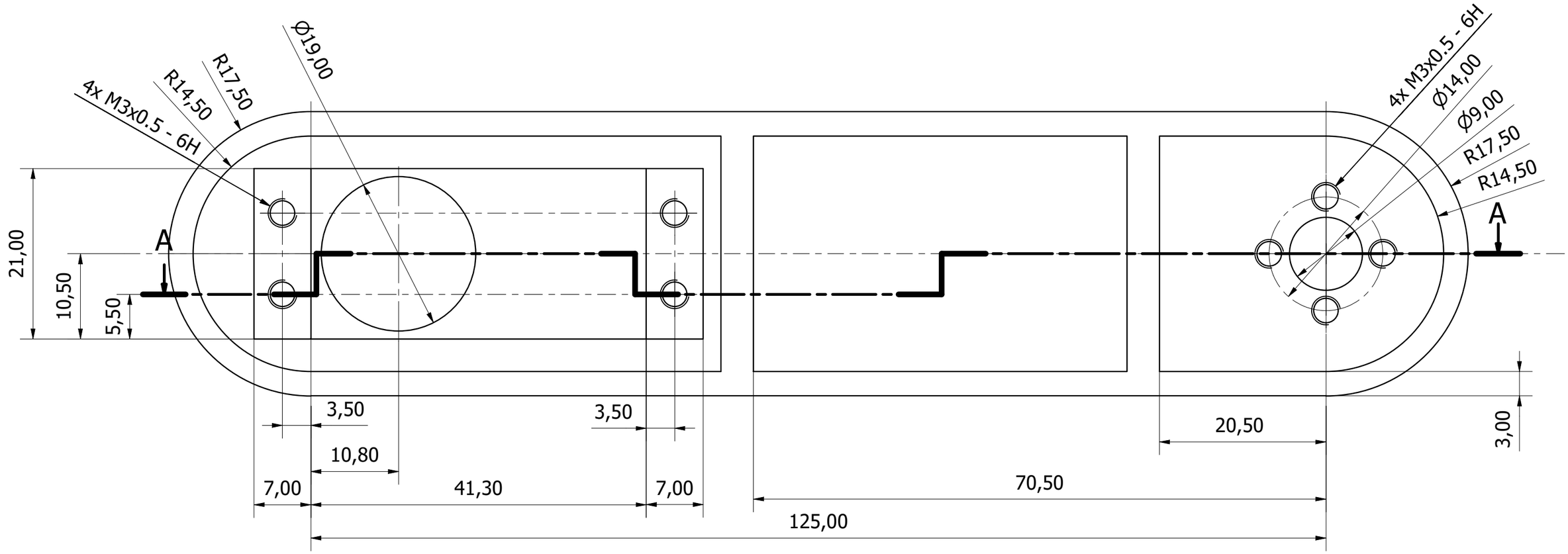
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	

	Przesuw		
Rysunek techniczny		Wydanie 1	Arkusz 1 / 1

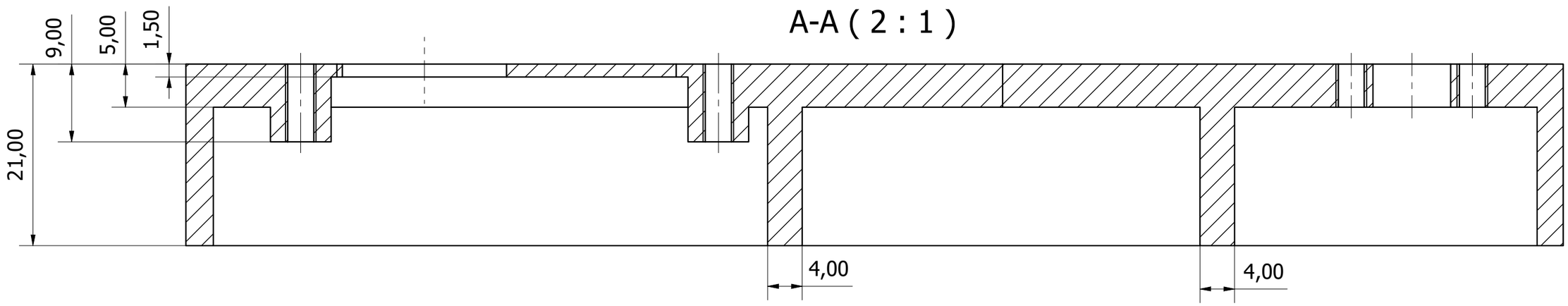
LISTA CZĘŚCI		
POZYCJA	ILOŚĆ	NUMER CZĘŚCI
1	1	Ramie 1
2	1	MG996R Serwo
3	1	Orczyk
4	4	Śruba M3x12
5	5	Śruba M3x8



Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data	Data	
Marcin Okrój				05.12.2022	
			Ramie 1		
			Rysunek złożeniowy	Wydanie 1	Arkusz 1 / 1

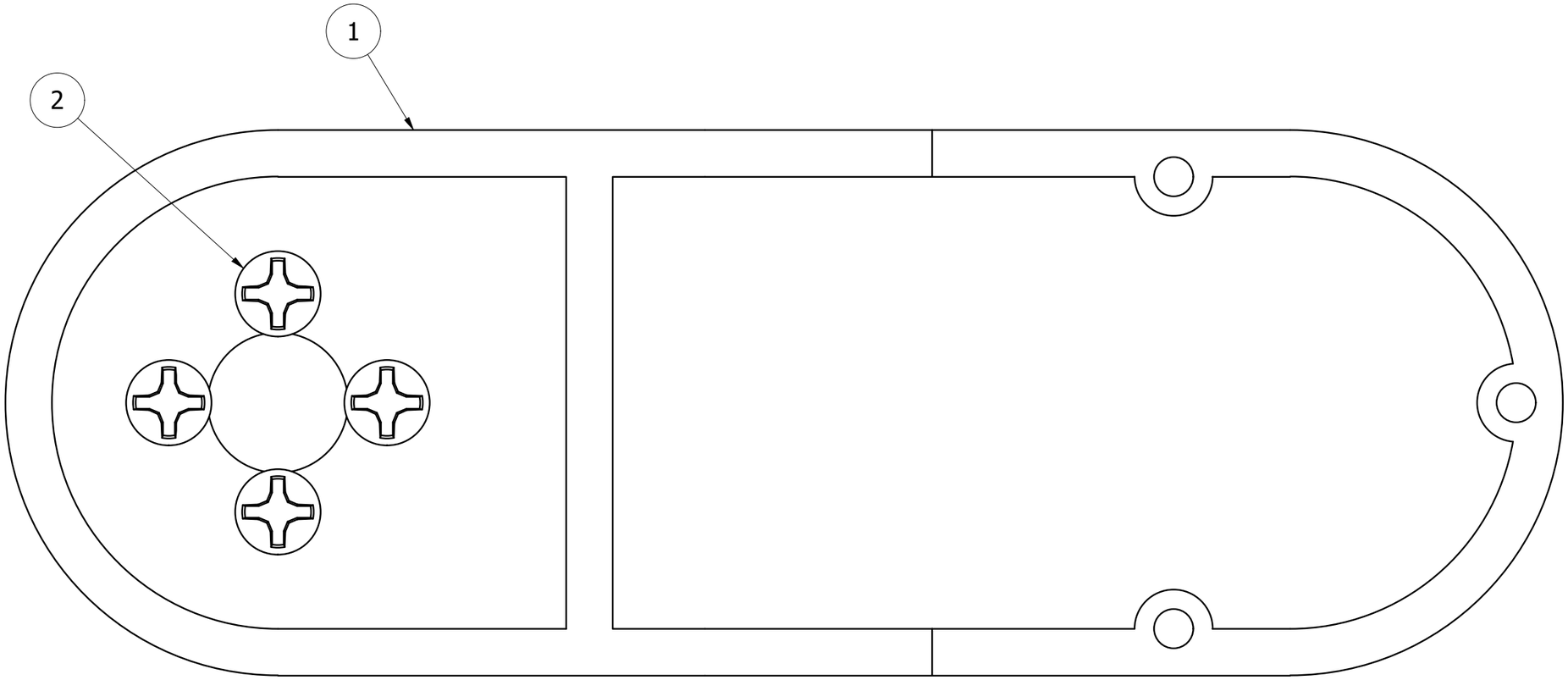


A-A (2 : 1)



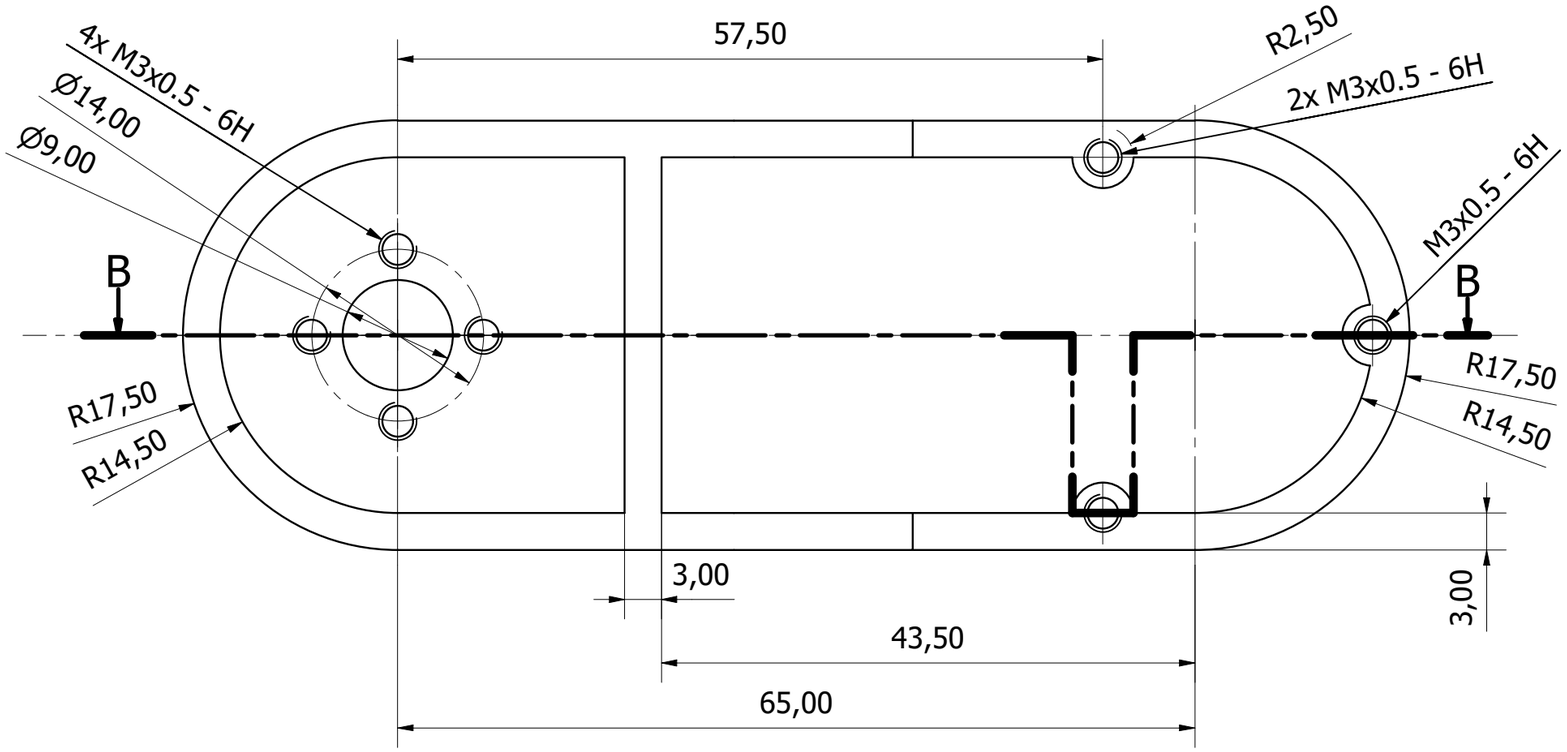
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data	Data	
Marcin Okrój				05.12.2022	

	Ramie 1		
Rysunek techniczny	Wydanie 1	Arkusz 1 / 1	

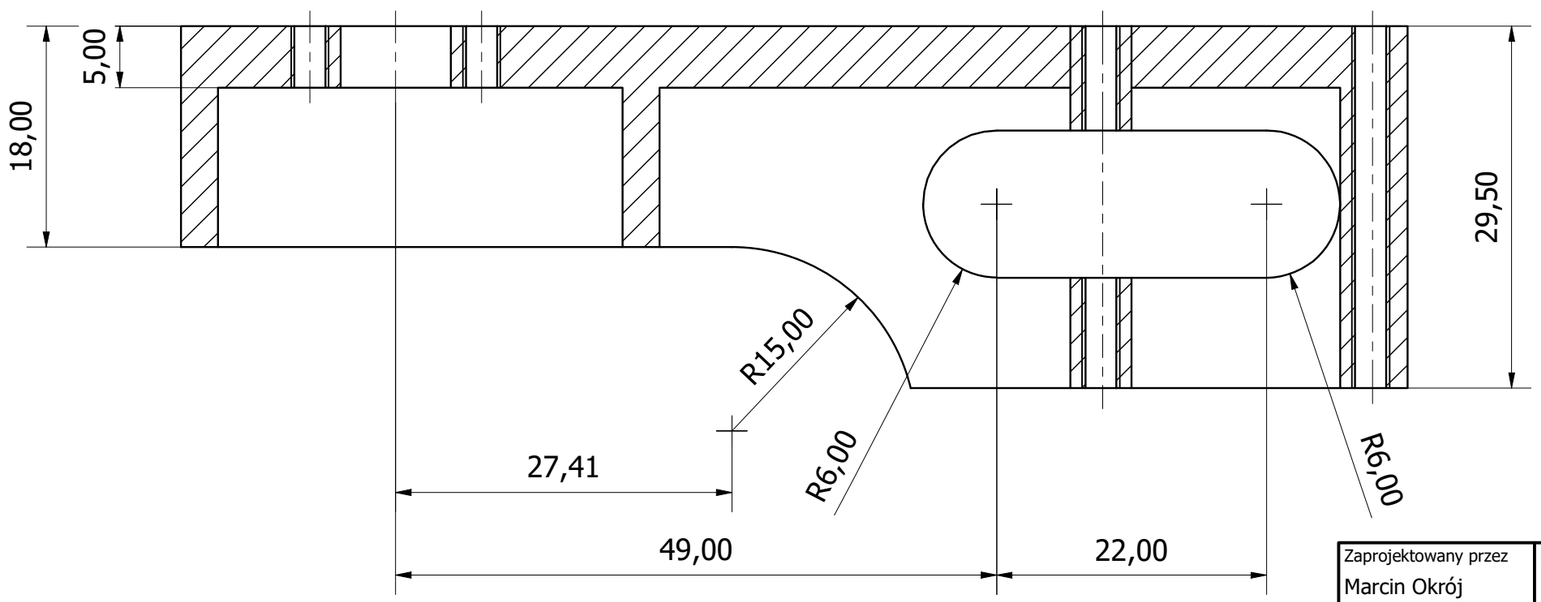


LISTA CZĘŚCI		
POZYCJA	ILOŚĆ	NUMER CZĘŚCI
1	1	Ramie 2
2	4	Śruba M3x8

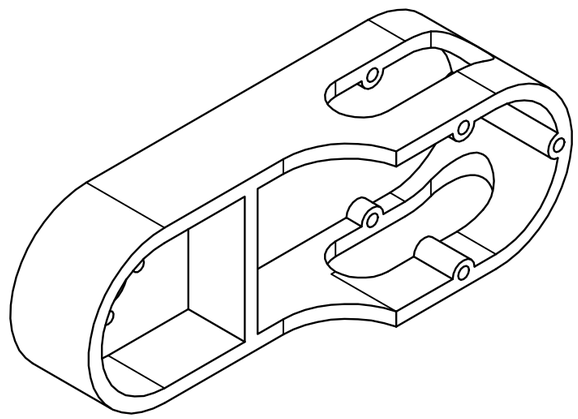
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	
				Ramie 2		
				Rysunek złożeniowy	Wydanie 1	Arkusz 1 / 1



B-B (2:1)



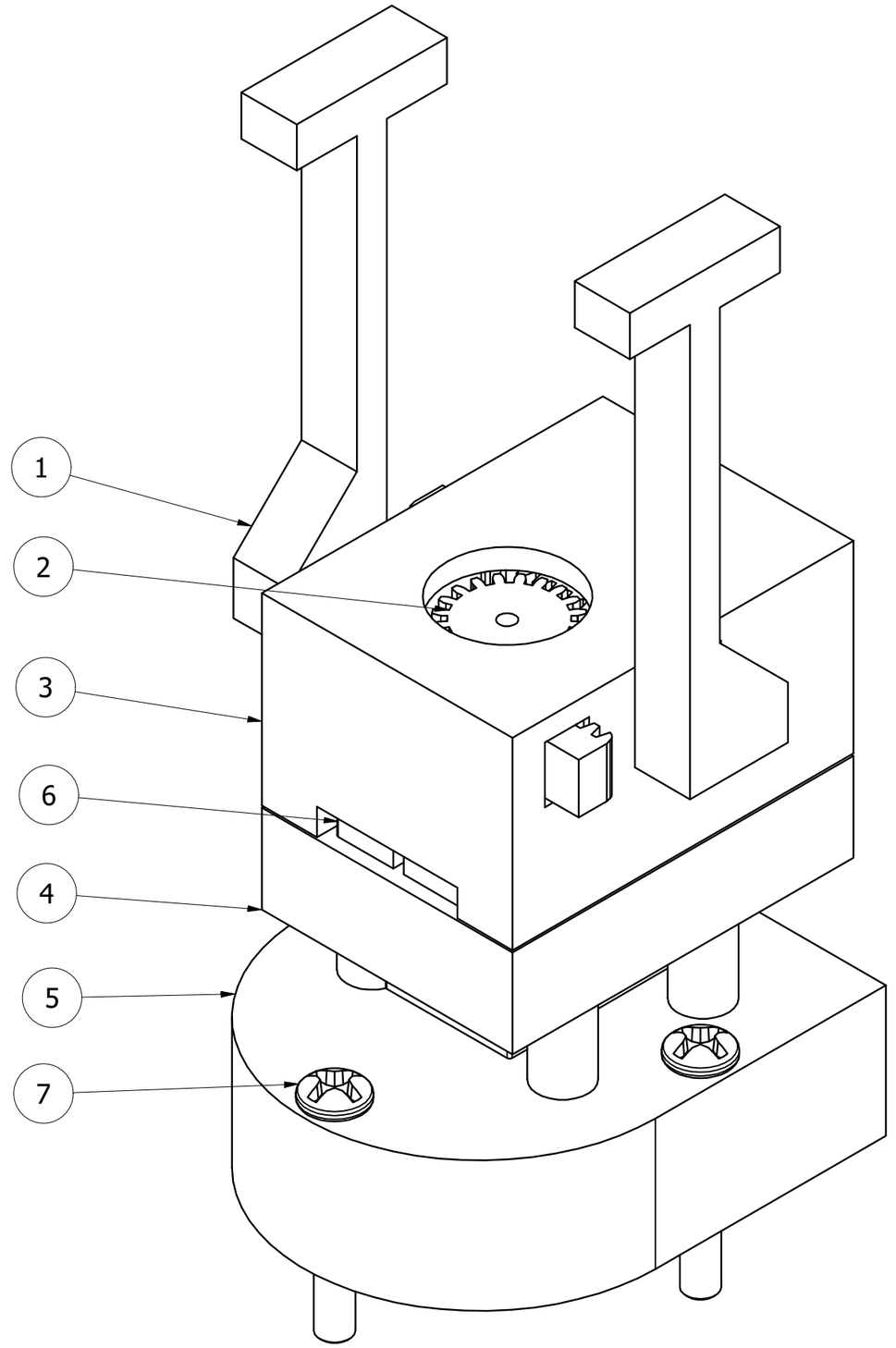
(1:1)



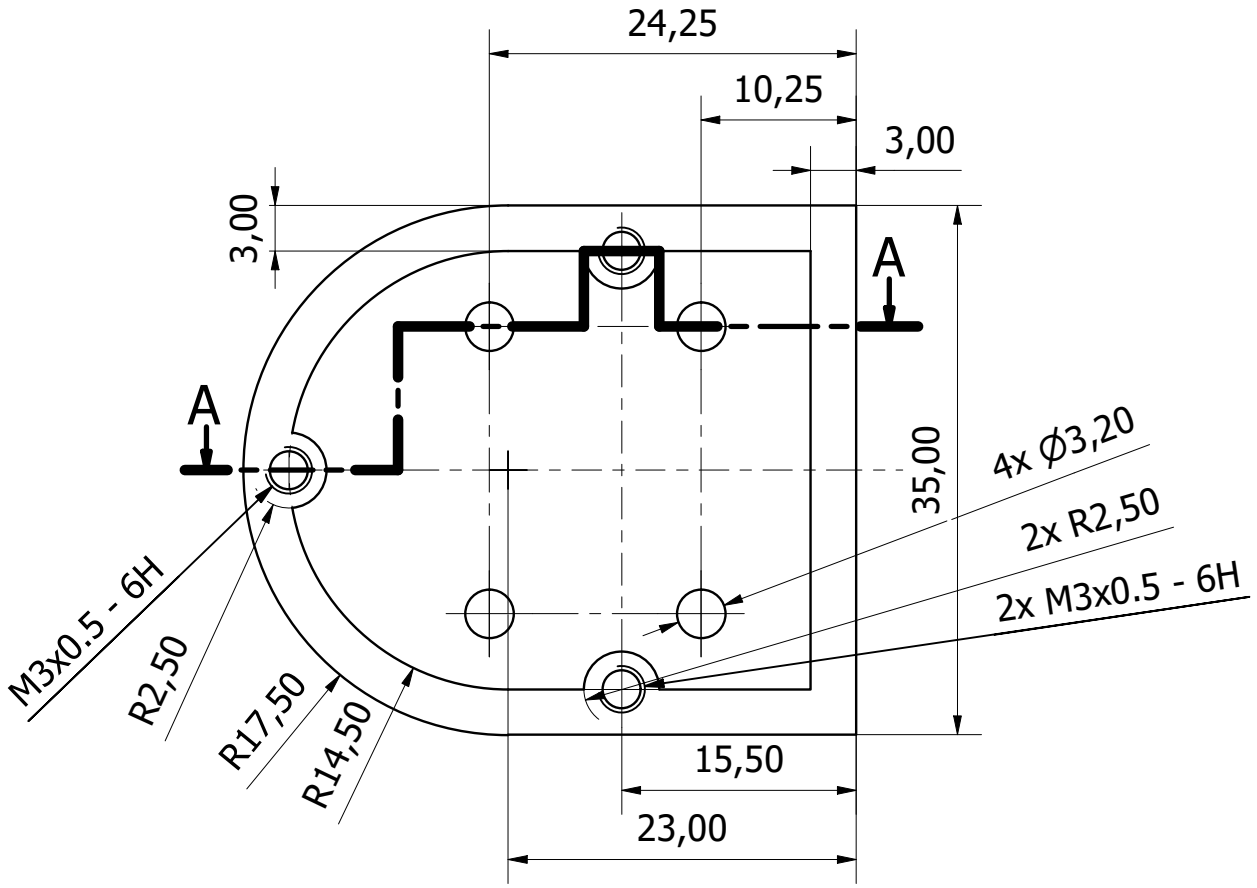
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	

	Ramie 2		
	Rysunek techniczny	Wydanie 1	Arkusz 1 / 1

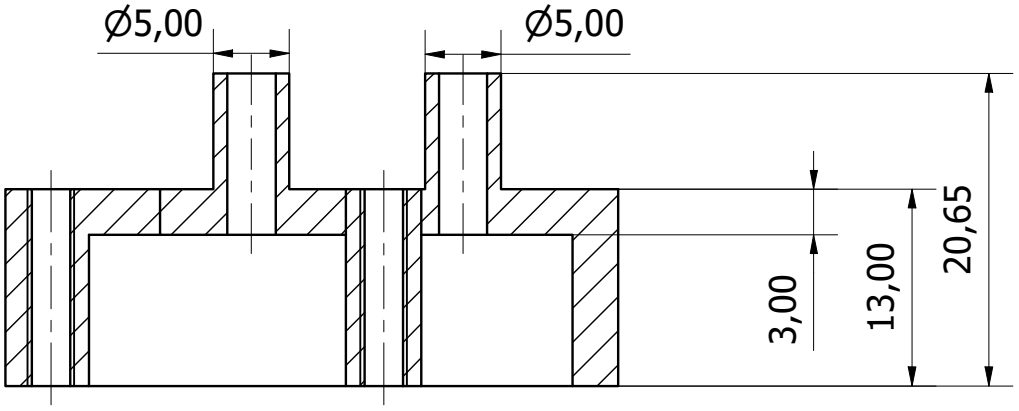
LISTA CZĘŚCI		
POZYCJA	ILOŚĆ	NUMER CZĘŚCI
1	2	Chwytak ramie
2	1	Chwytak zębatka
3	1	Obudowa
4	1	Środek
5	1	Chwytak mocowanie
6	1	SG90 Mikro serwo
7	3	Śruba M3x20



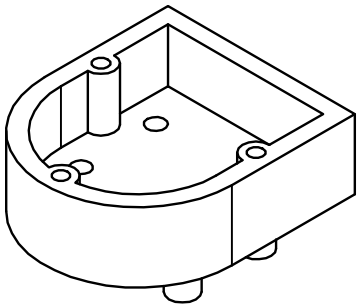
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	
				Chwytak		
				Rysunek złożeniowy	Wydanie 1	Arkusz 1 / 1



A-A (2:1)

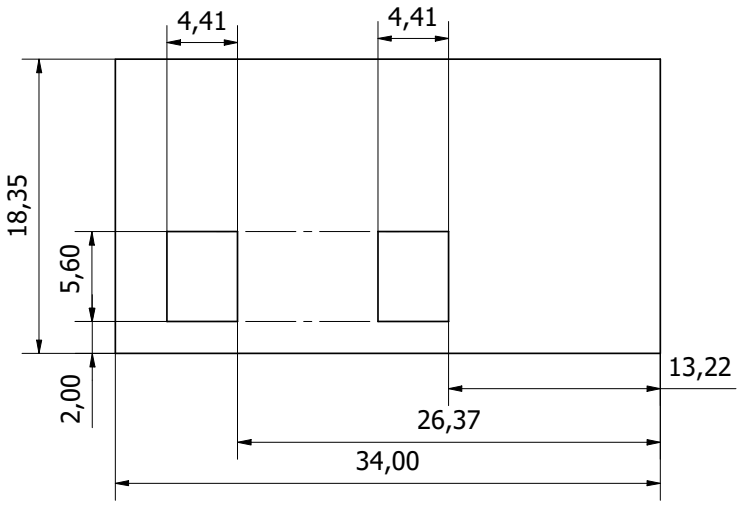


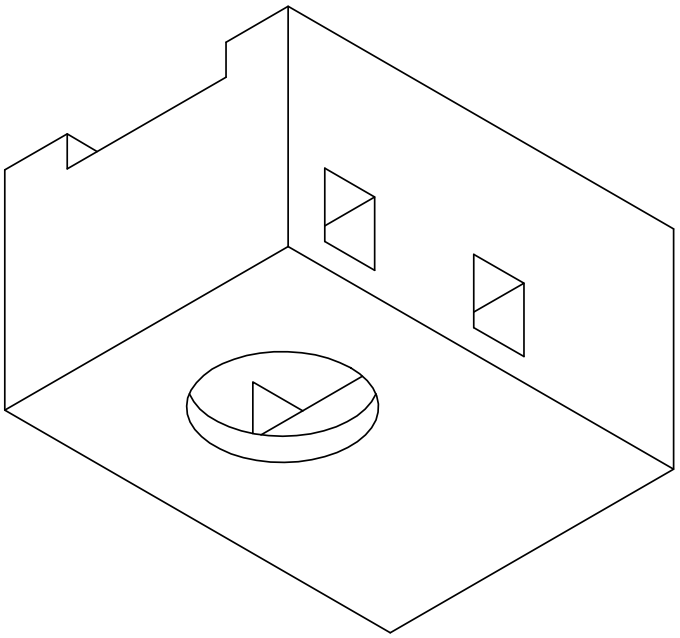
(1:1)

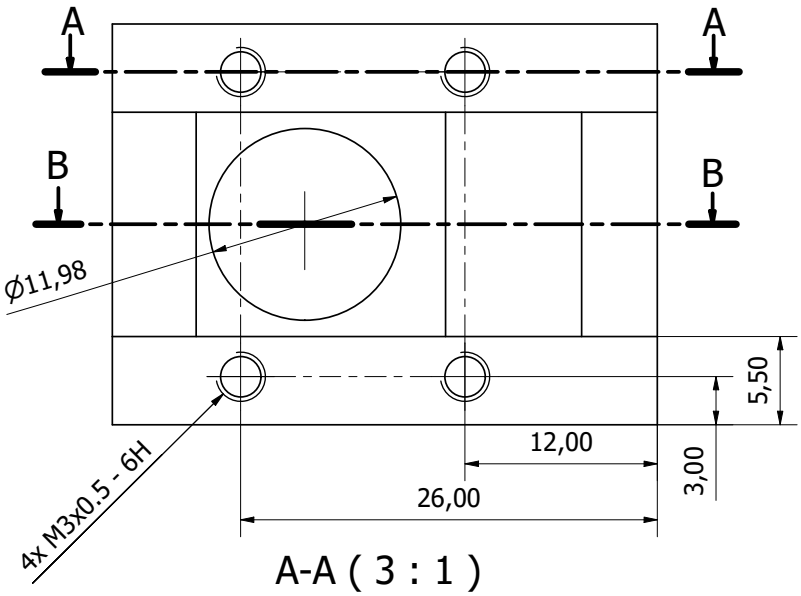


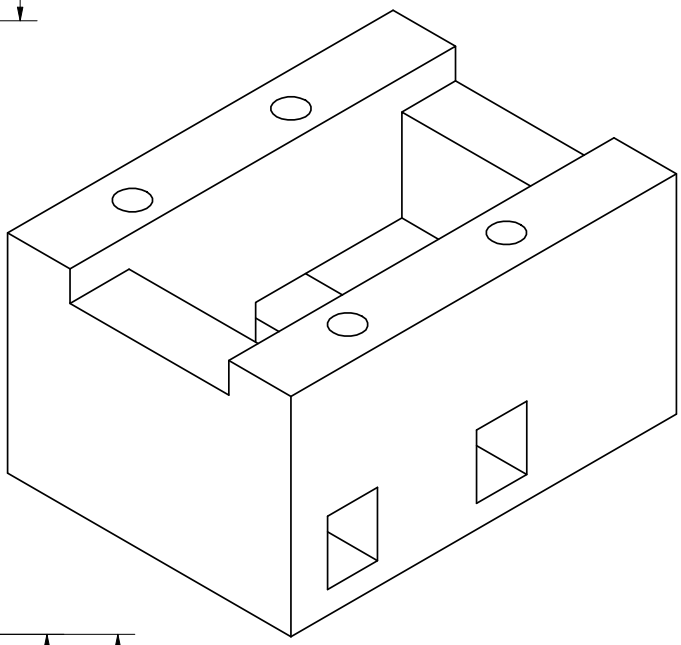
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	

	Chwytak mocowanie		
	Rysunek techniczny	Wydanie 1	Arkusz 1 / 1

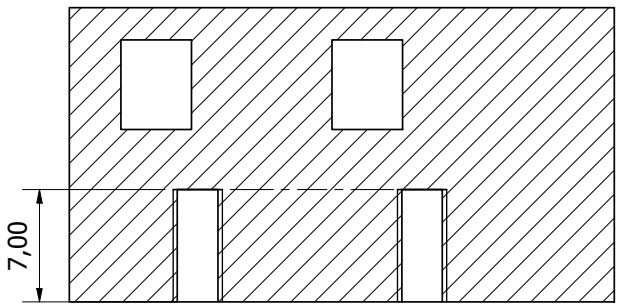




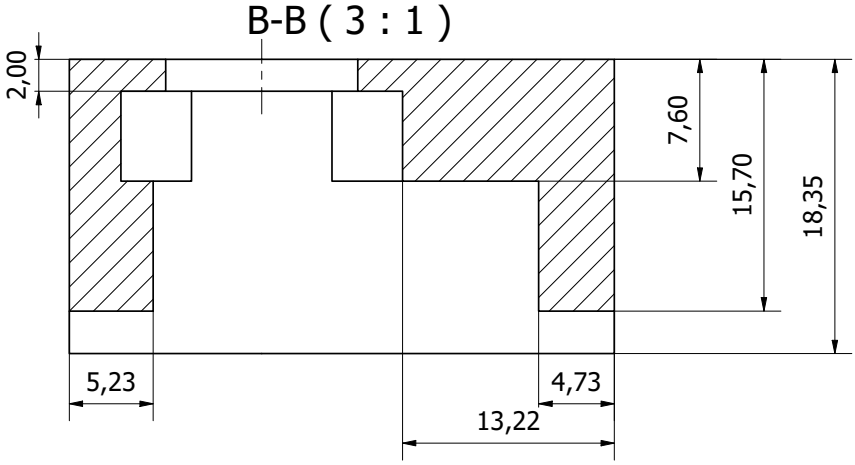




A-A (3 : 1)

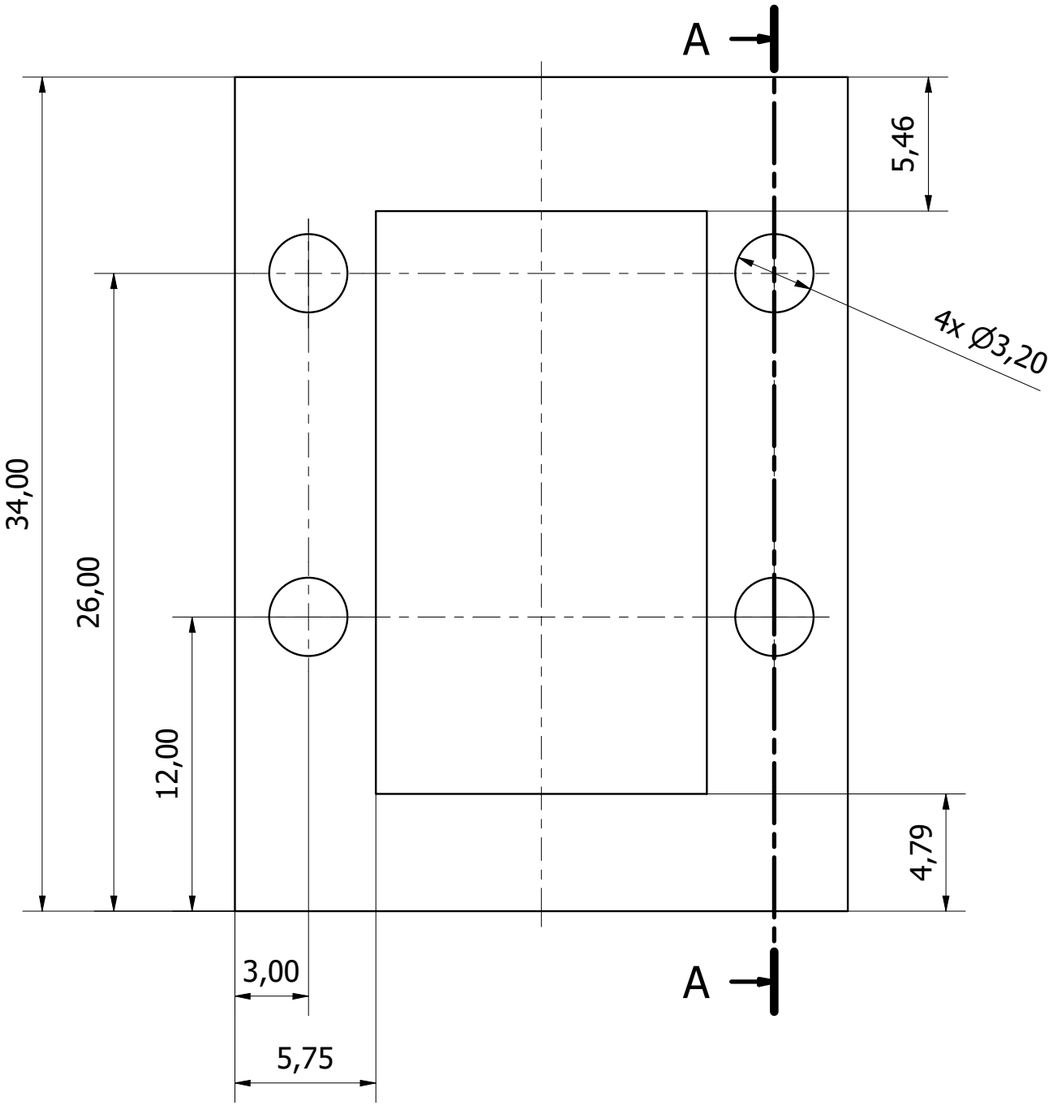


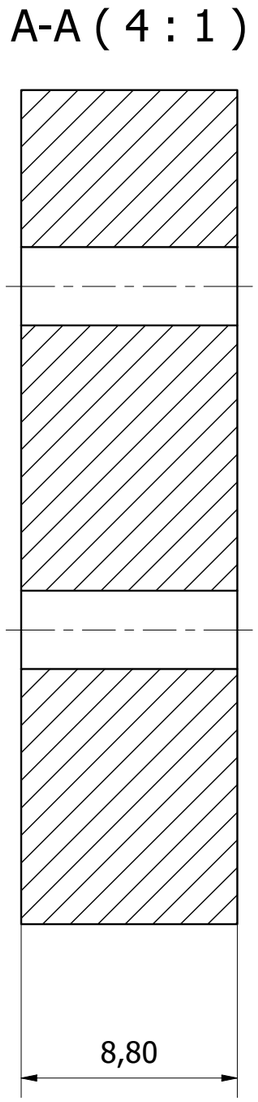
B-B (3 : 1)

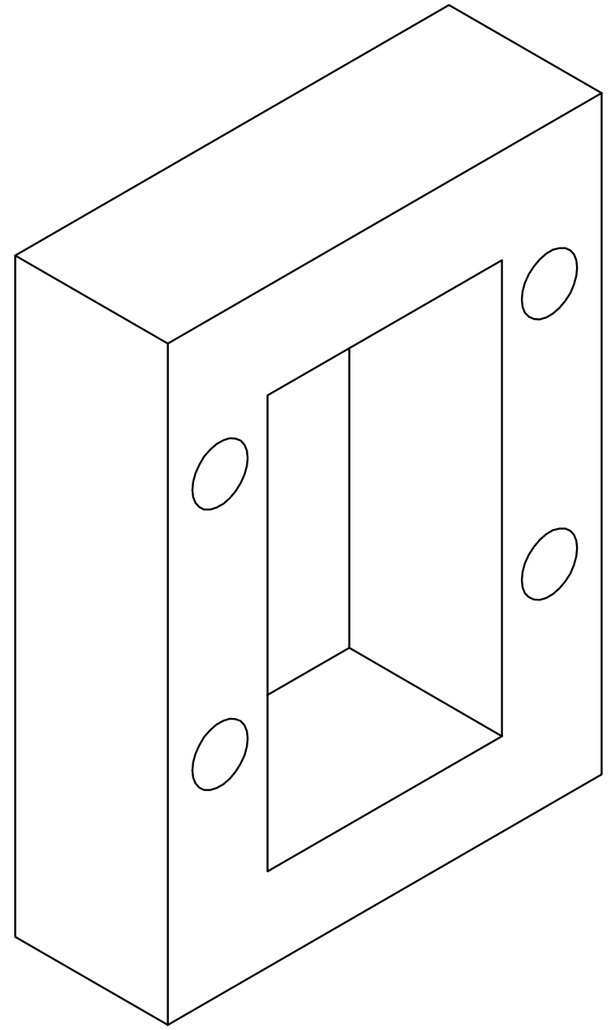


Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	

	Chwytak obudowa		
Rysunek techniczny		Wydanie 1	Arkusz 1 / 1

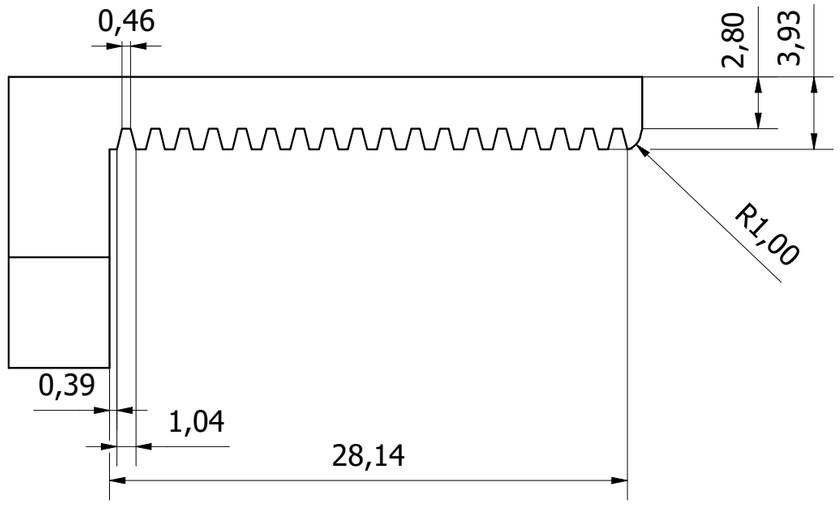


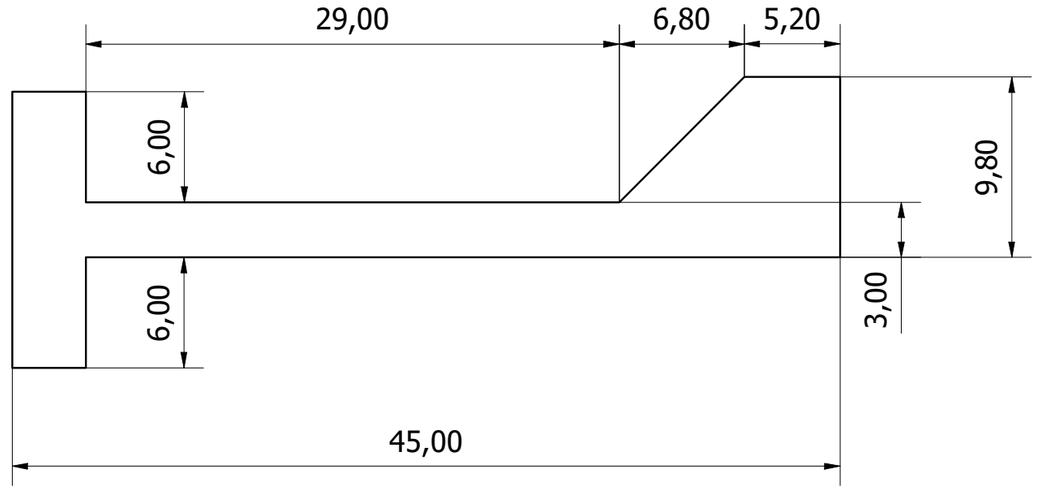


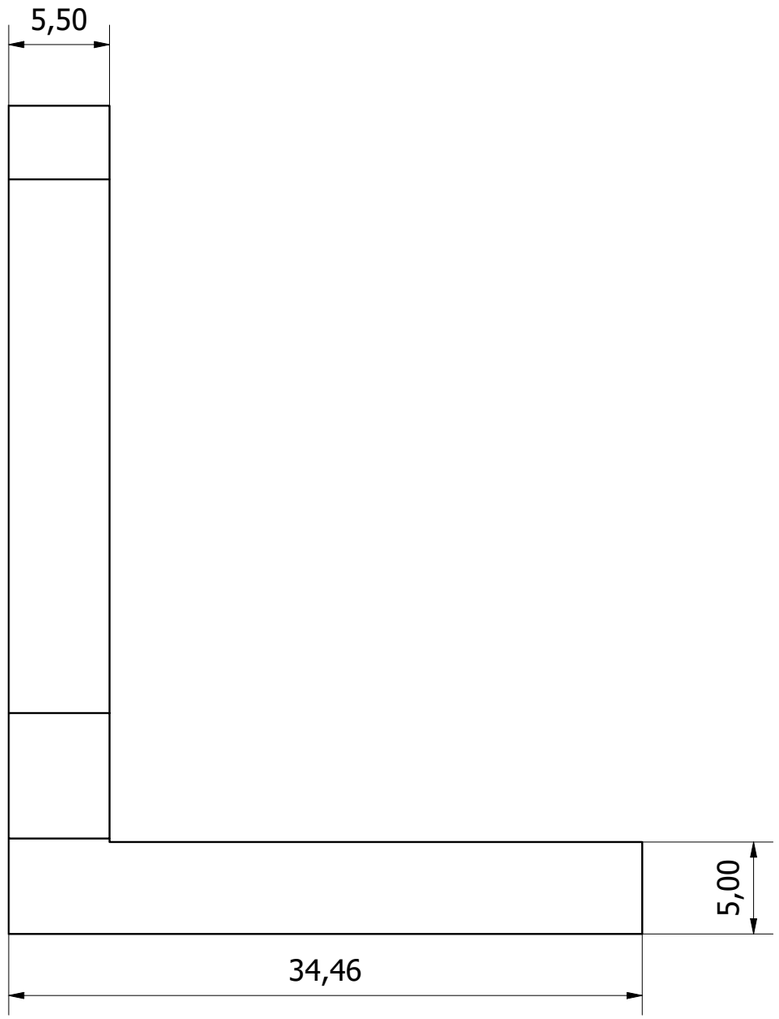


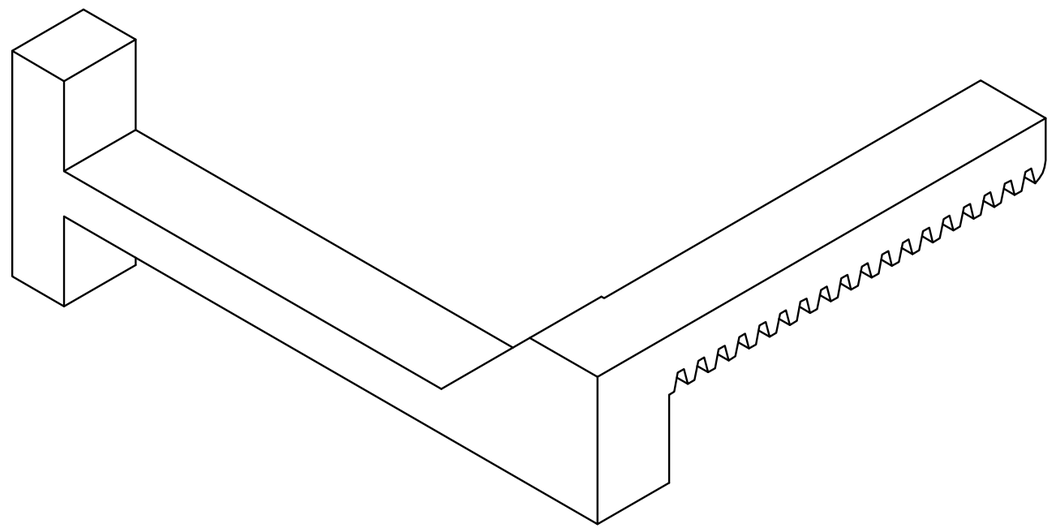
Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	

	Chwytak środek		
Rysunek techniczny		Wydanie 1	Arkusz 1 / 1









Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data	Data	
Marcin Okrój				05.12.2022	

	Chwytak ramie		
	Rysunek techniczny	Wydanie 1	Arkusz 1 / 1

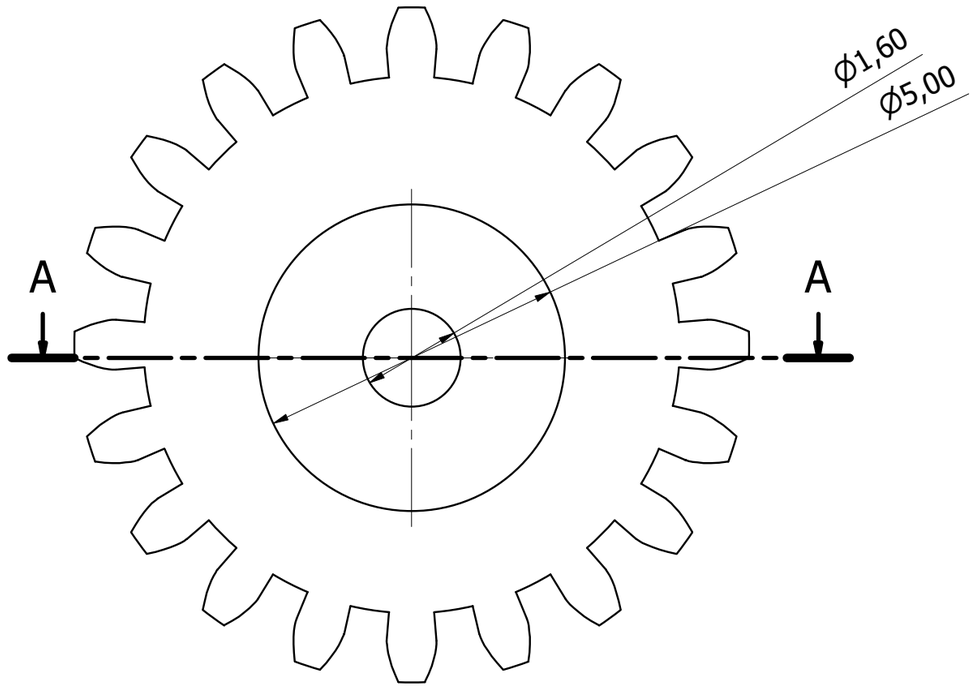
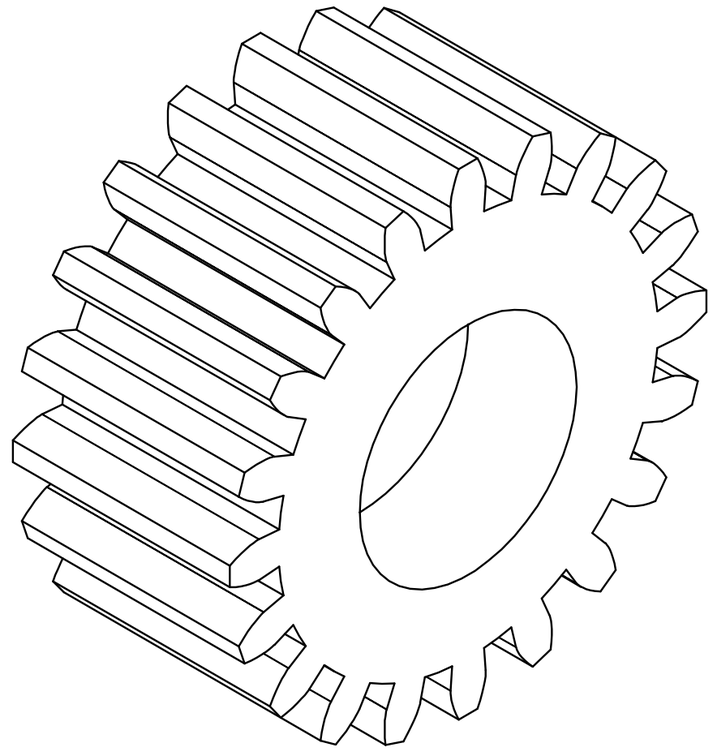
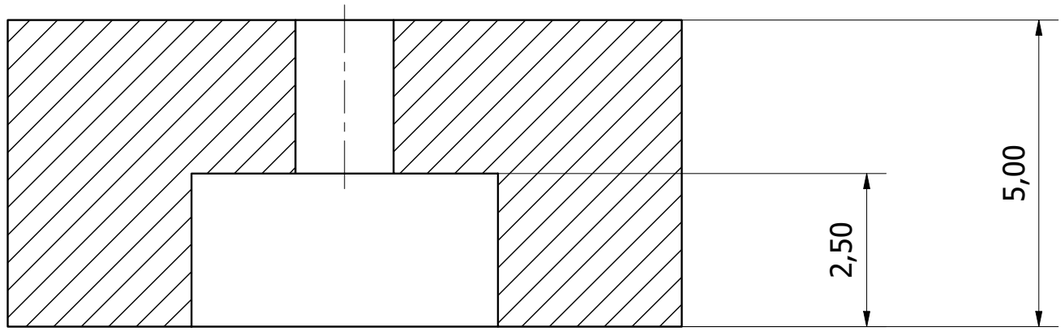


TABELA	
Kolumna 1	Kolumna 2
Moduł	0,5 mm
Liczba zębów	20
Kąt przyporu	14,5°



A-A (10 : 1)



Zaprojektowany przez	Sprawdzony przez	Zatwierdzony przez	Data		Data	
Marcin Okrój					05.12.2022	

	Chwytak zębatka		
	Rysunek techniczny	Wydanie 1	Arkusz 1 / 1