Projekt Zespołowy
Budowa robota sześcioosiowego z systemem wizyjnym

Autorzy:
Patryk Wieczorek
Michał Prośba
Jakub Magdziarek

Prowadzący:
Dr. inż. Andrzej Rusiecki
Mgr. inż. Cyprian Mataczyński

9 czerwca 2021

1 Założenia projektowe

Celem naszego projektu jest zbudowanie robota sześcioosiowego wyposażonego w system wizyjny. Manipulator ma za zadanie sortować klocki na podstawie kształtu ich górnej ścianki.

2 Etapy projektu

1. Dobór i zakup układów elektronicznych oraz mikrokontrolera pozwalającego na sterowanie manipulatorem.
2. Dobór i zakup części mechanicznych (łożyska, osie).
3. Zaprojektowanie komponentów robota w programie Inventor.
4. Wydrukowanie części robota sześcioosiowego za pomocą drukarek 3D.
5. Oprogramowanie podstawowych ruchów robota - STM32F446RE.
6. Oprogramowanie systemu wizyjnego - ESP32CAM.
7. Oprogramowanie Web Servera odpowiedzialnego za przekazywanie informacji zwrotnych do robota - NodeMCU.
8. Znalezienie odpowiedniego zbioru danych do nauki sieci neuronowej.
9. Stworzenie modelu sieci neuronowej do rozpoznawania obiektów - Python Tensorflow.
10. Nauka i testowanie sieci neuronowej
11. Znalezienie i zaimplementowanie sposobu komunikacji sieci z robotem esp32-kamera ⇒ komputer ⇒ stm32-robot
12. Integracja wszystkich segmentów - finalizacja projektu.

3 Część Mechaniczna

Realizacja tej części wymagała ogromnego nakładu czasu. Po zamówieniu i zwymiarowaniu elektroniki nadszedł czas na zaprojektowanie potrzebnych części mechanicznych za pomocą programu Inventor. Po drodze napotkaliśmy wiele problemów związanych głównie z naciągami pasków oraz odpowiedniego montażu łożysk, co skutkowało wielokrotnymi poprawkami oraz potrzebą wielokrotnego drukowania części. Po dopracowaniu wszystkich szczegółów, udało się bez większych problemów złożyć manipulator i jednocześnie zakończyć ten etap.

4 Część Programistyczna - sieć neuronowa

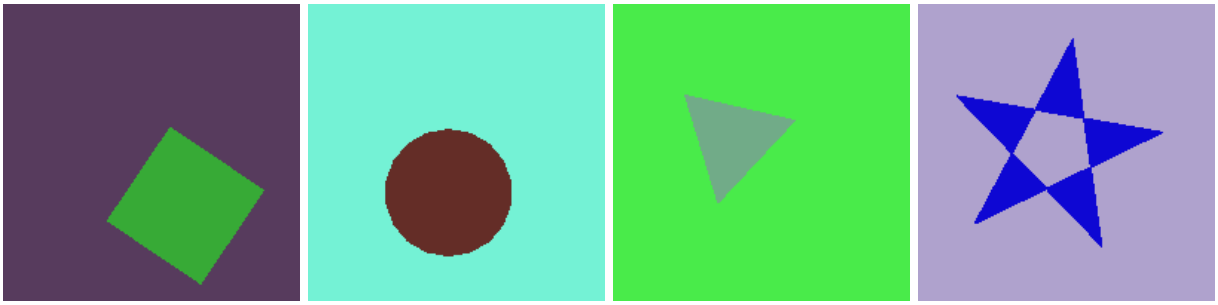
Do realizacji tej części potrzebne było znalezienie odpowiedniego zbioru danych przedstawiającego figury geometryczne. Udało się znaleźć zbiór z 40000 elementami przedstawiającymi koło, kwadrat, gwiazde i trójkąt w różnych kolorach różnej wielkości i ułożone pod różnym kątem. Nastepnie został stworzony model konwolucyjnej sieci neuronowej przy wykorzystaniu pakietu tensorflow-keras. Do uczenia i przetestowania sieci zbiór został podzielony na 3 części:

1. uczącą - 16000 elementów
2. walidacyjną - 8000 elementów
3. testową- 16000 elementów

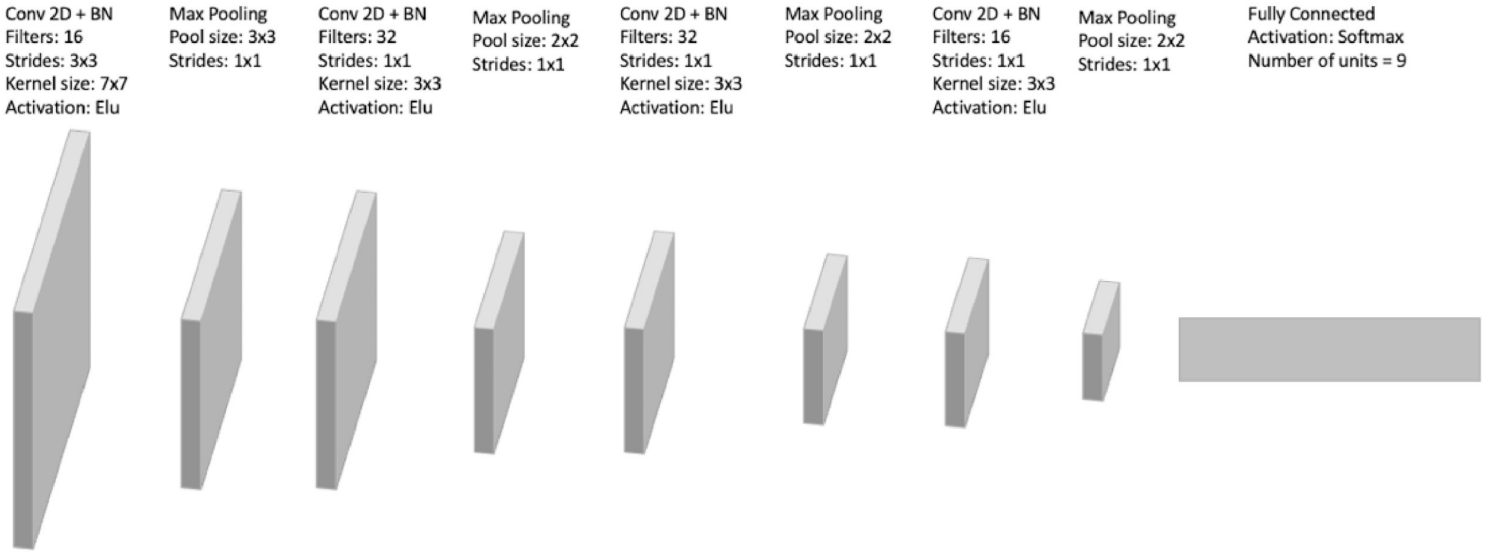
Parametry przy uczeniu:

1. batch size = 50
2. epochs = 26
3. steps per epoch = 720

Sieć uzyskała dokładność 98% dla zbioru testowego. Model sieci składa się z 9 warstw pokazanych na rysunku 2



Rysunek 1: Przykładowe obrazy ze zbiory danych do uczenia sieci neuronowej



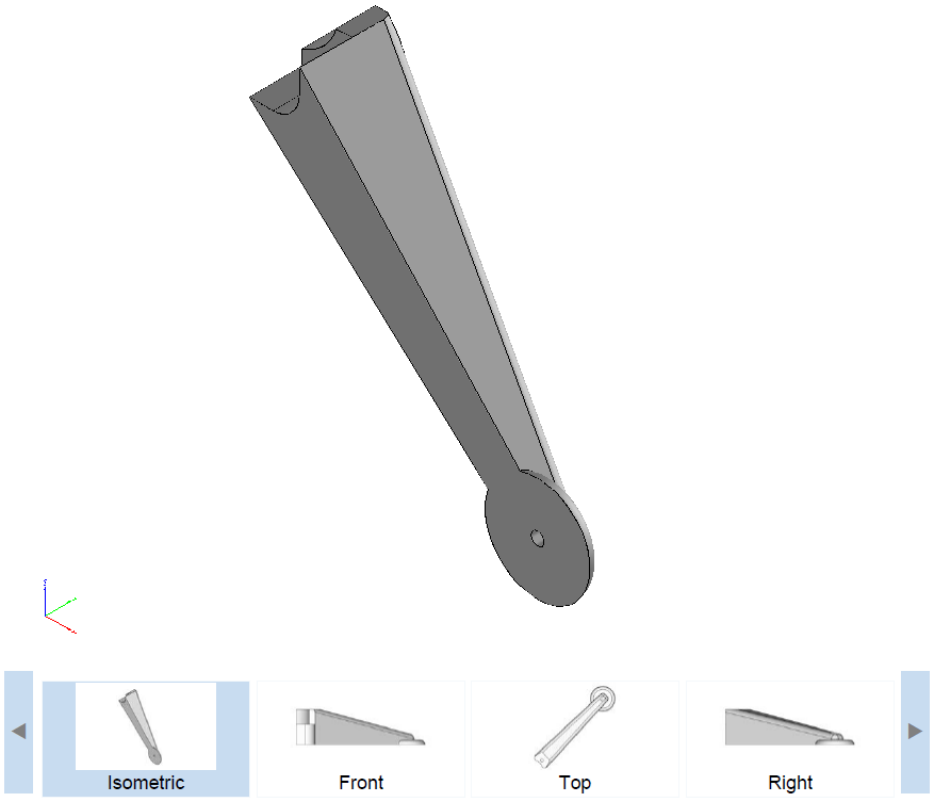
Rysunek 2: Grafika przedstawiająca model sieci neuronowej

5 Część Programistyczna - oprogramowanie mikrokontrolerów

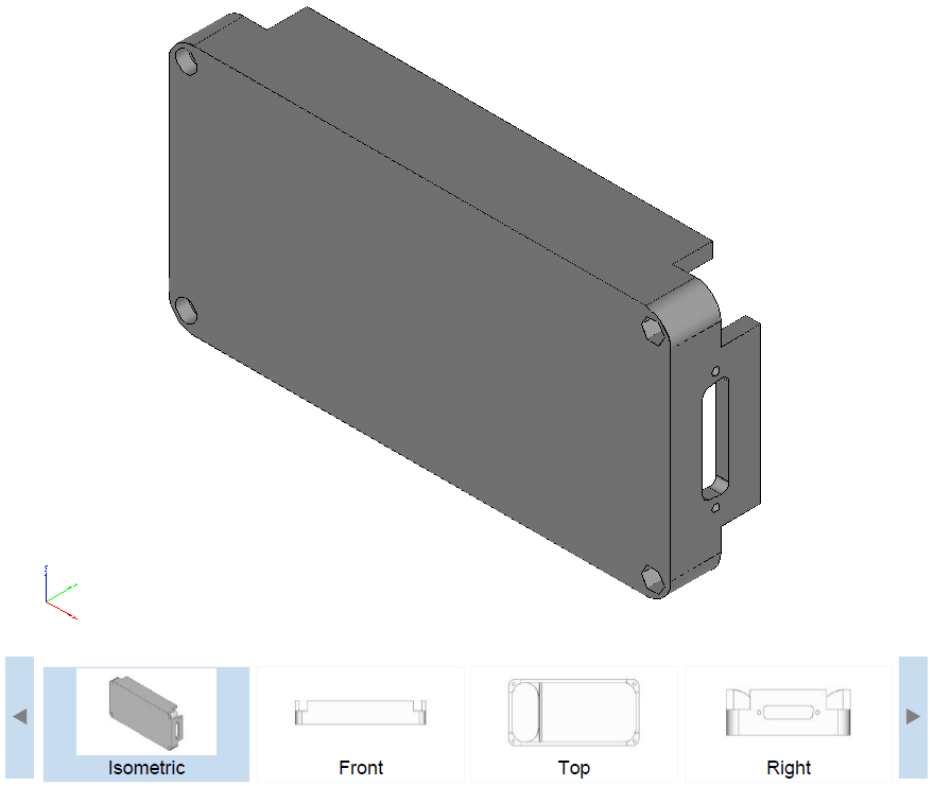
Po przygotowaniu części mechanicznej oraz napisaniu sieci neuronowej, nadszedł czas na integrację poprzednich etapów za pośrednictwem mikrokontrolerów ESP32CAM oraz NodeMCU. ESP32CAM zostało skonfigurowane w taki sposób aby wystawiać na adres 192.168.1.150 obraz z kamery. Program napisany w pythonie, pobiera obraz z wcześniej podanego adresu IP oraz rozpoznaje kształt klocka widocznego na zdjęciu za pomocą sieci neuronowej. Po rozpoznaniu kształtu wysyła sygnał do Web Servera skonfigurowanego na NodeMCU, który wystawia stan wysoki na odpowiednim wyjściu połączonym z STM32. Ostatnim już krokiem jest odebranie sygnału przez STM32 i wykonanie odpowiedniej sekwencji ruchów manipulatora (wrzucenie klocka do odpowiedniego pojemnika). Po wrzuceniu klocka manipulator wraca do pozycji domowej i czeka na kolejny sygnał.

6 Repozytorium

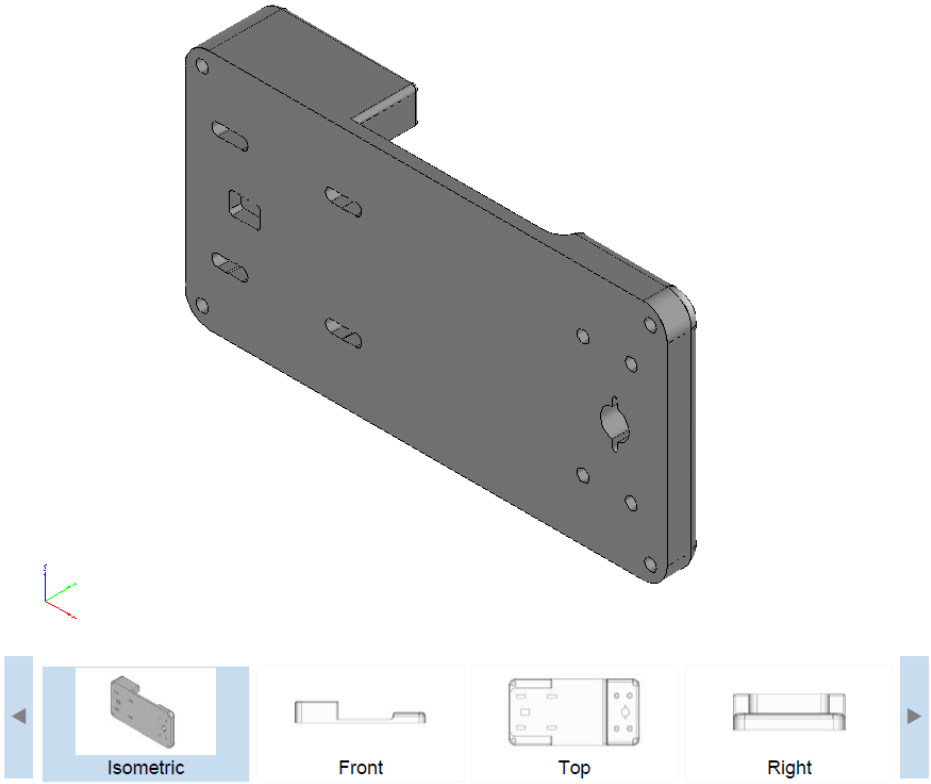
https://github.com/patrykwieczorek03/RobotSzescioosiowy



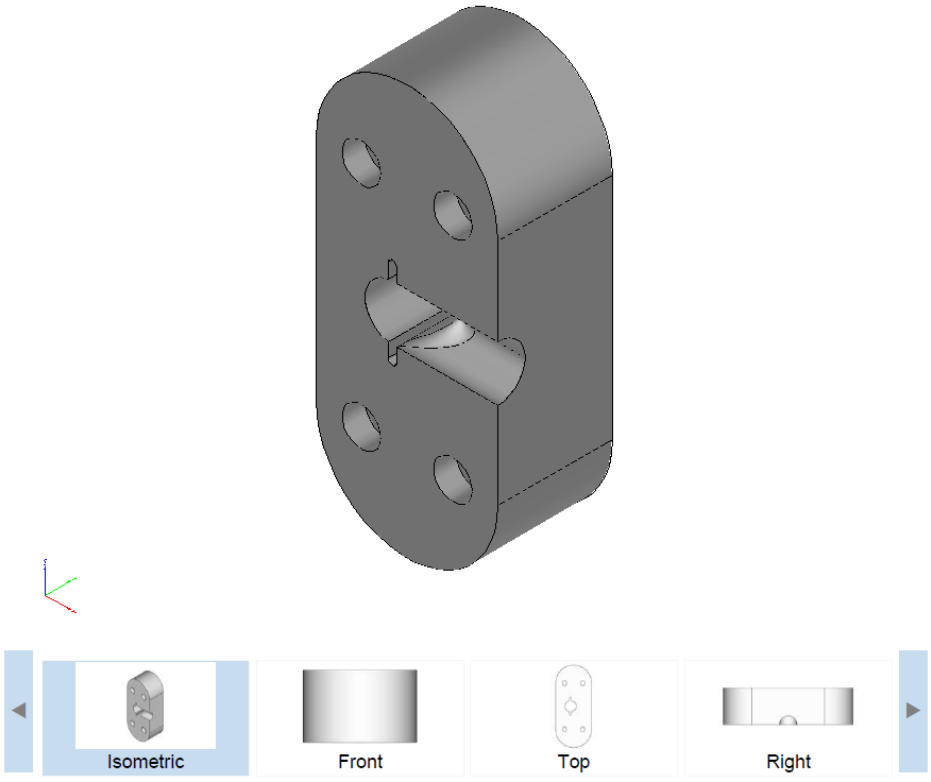
Rysunek 3: Noga



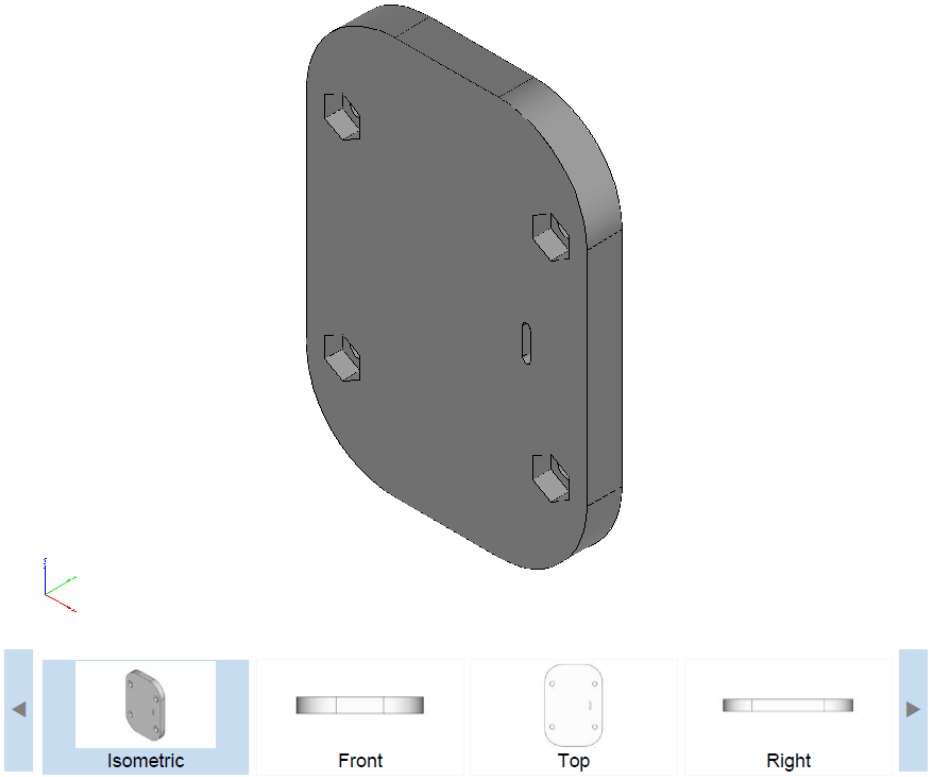
Rysunek 4: Podstawa



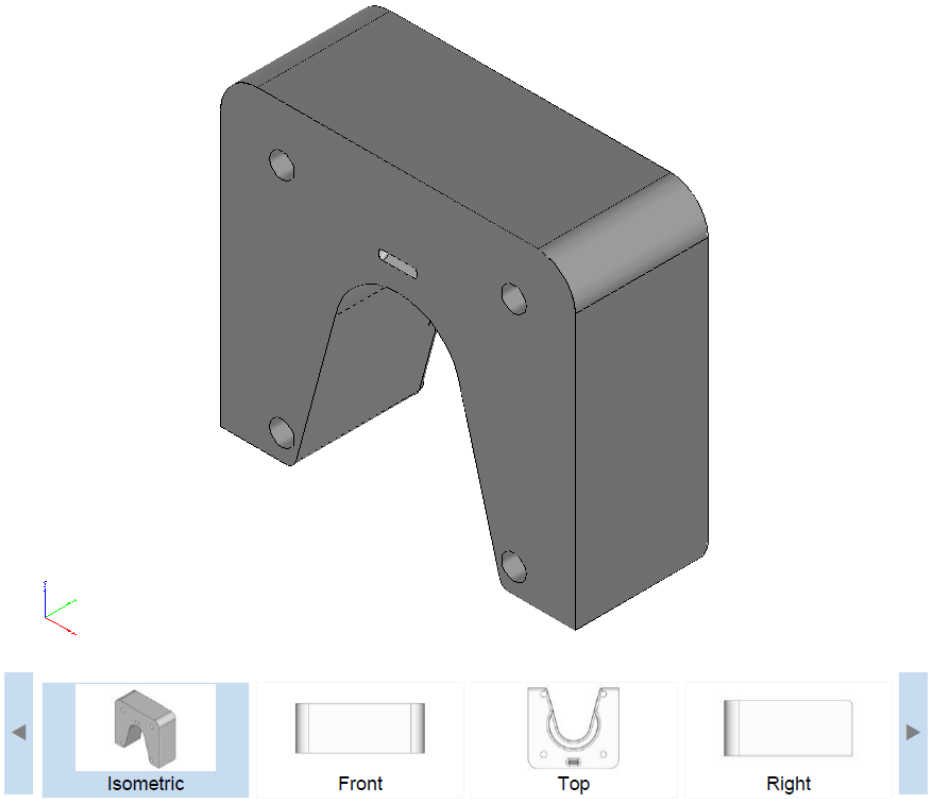
Rysunek 5: Pokrywka podstawy



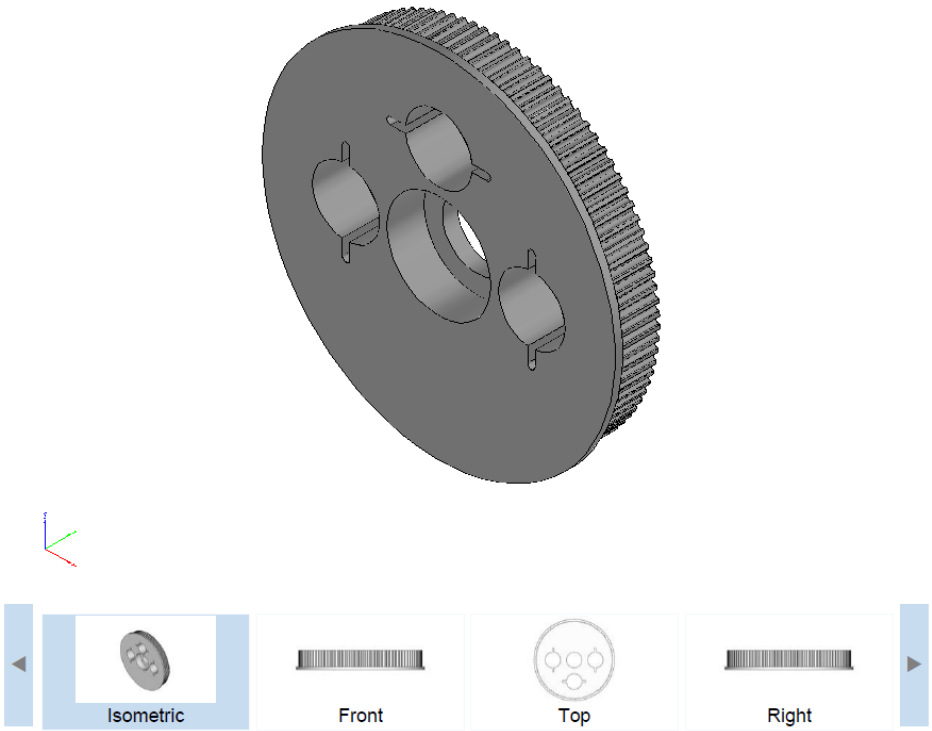
Rysunek 6: Mocowanie pierwszej osi



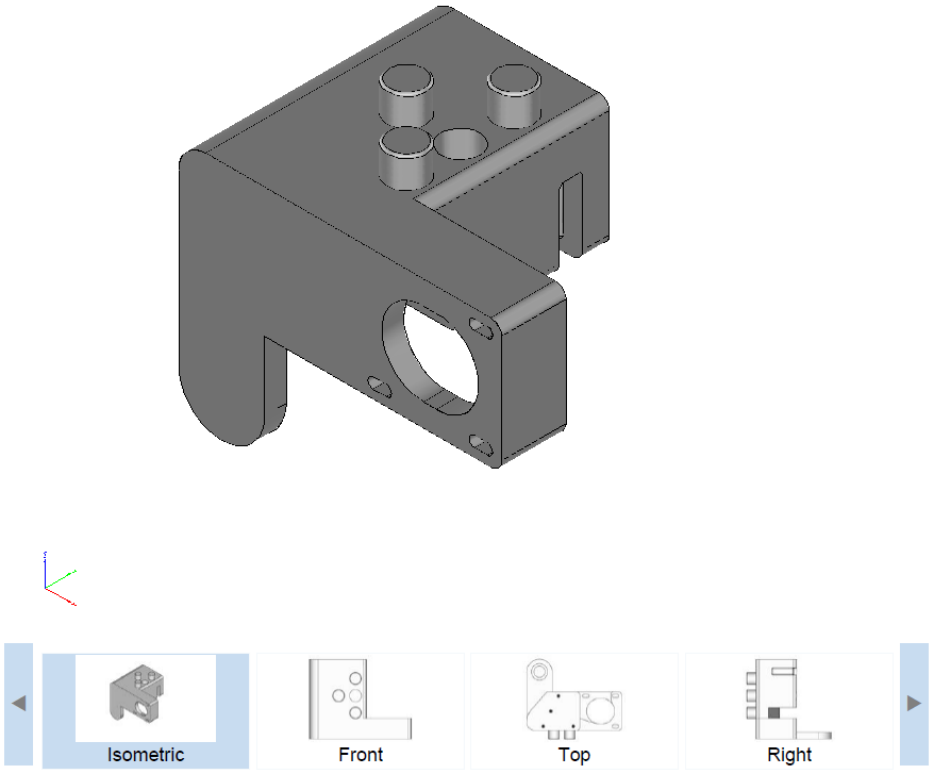
Rysunek 7: Część stabilizująca mocowanie pierwszego silnika



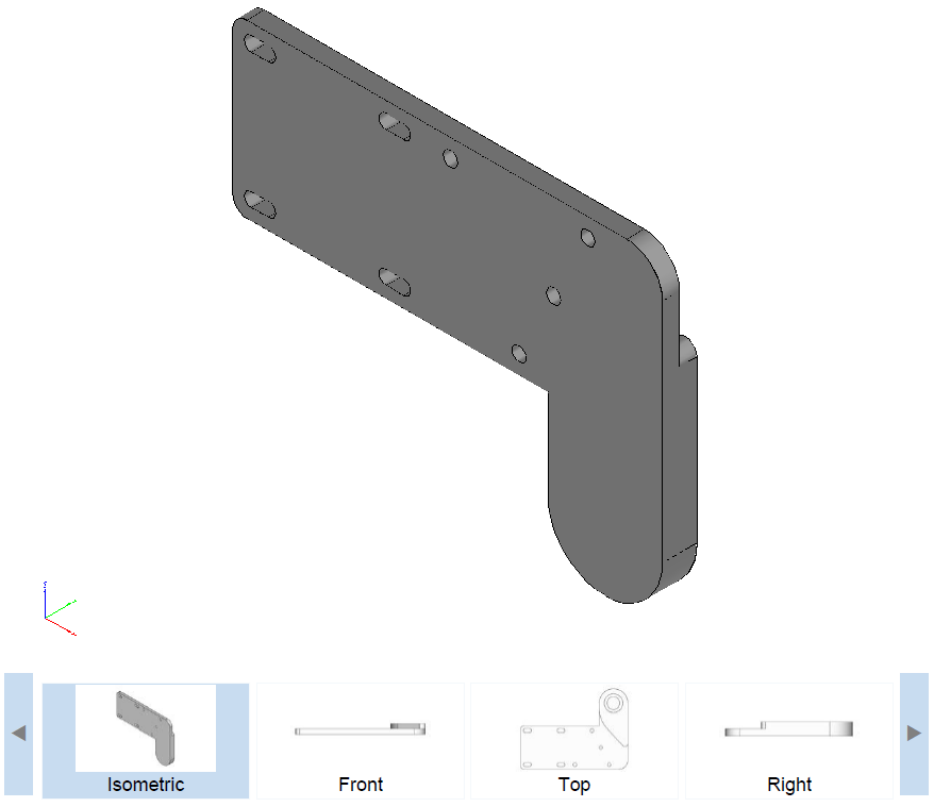
Rysunek 8: Mocowanie pierwszego silnika



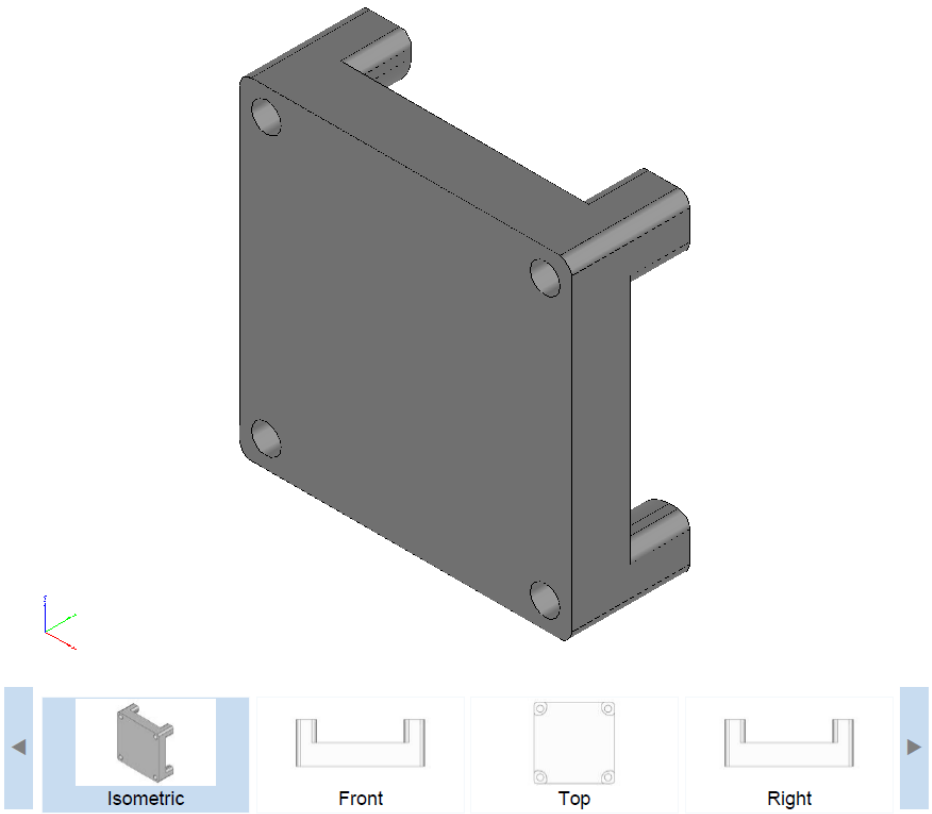
Rysunek 9: Zębatka pierwszej osi



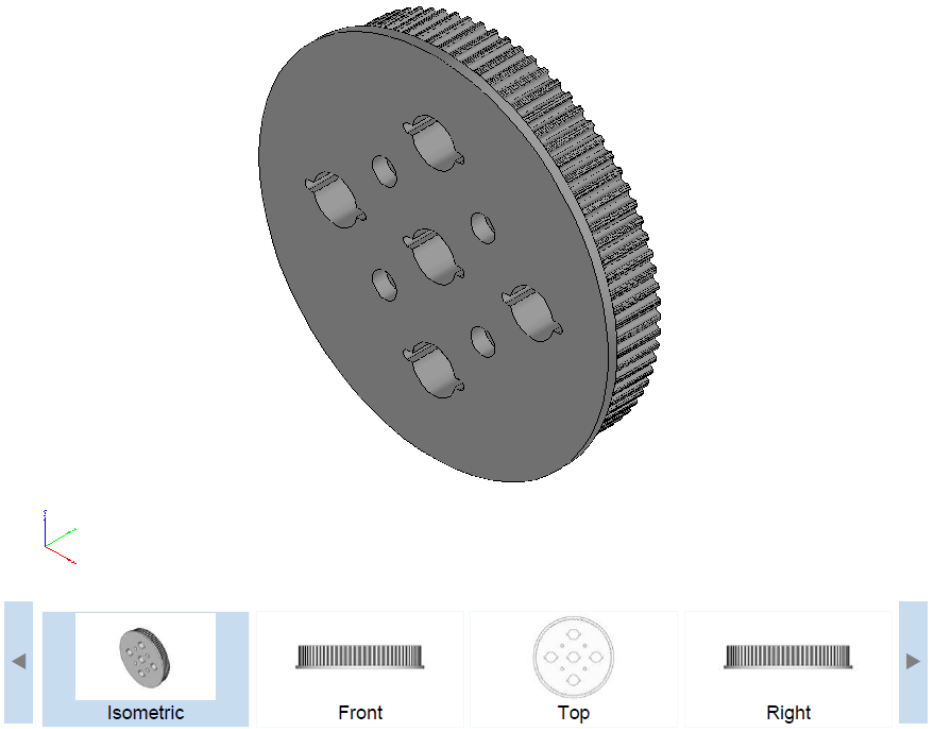
Rysunek 10: Mocowanie drugiego silnika



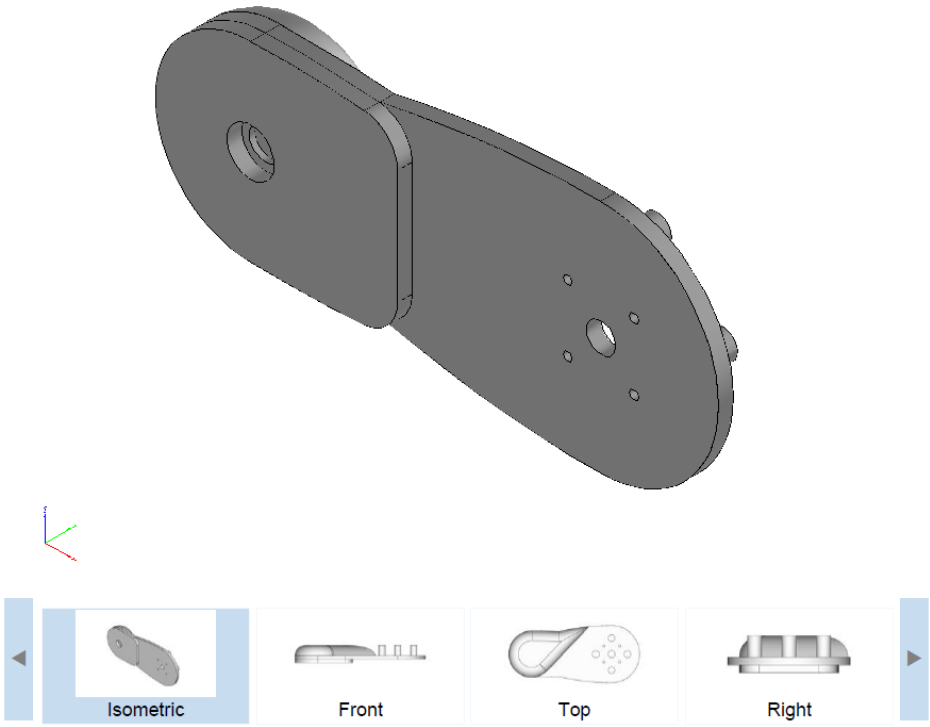
Rysunek 11: Mocowanie drugiegi silnika bok



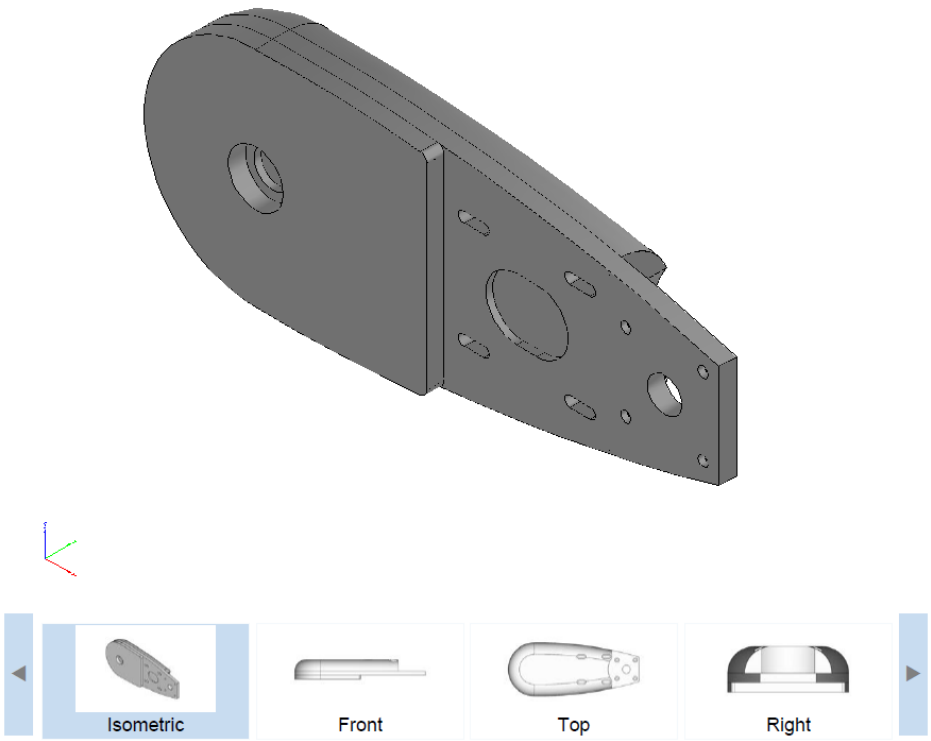
Rysunek 12: Tył mocowania drugiego silnika



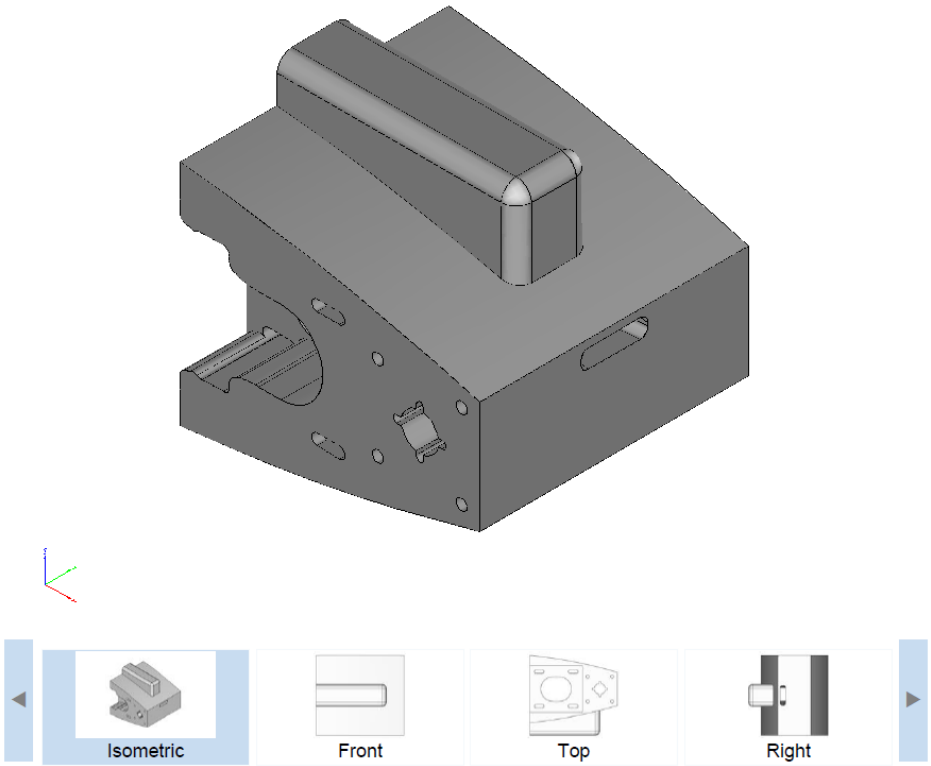
Rysunek 13: Zębatka drugiej osi



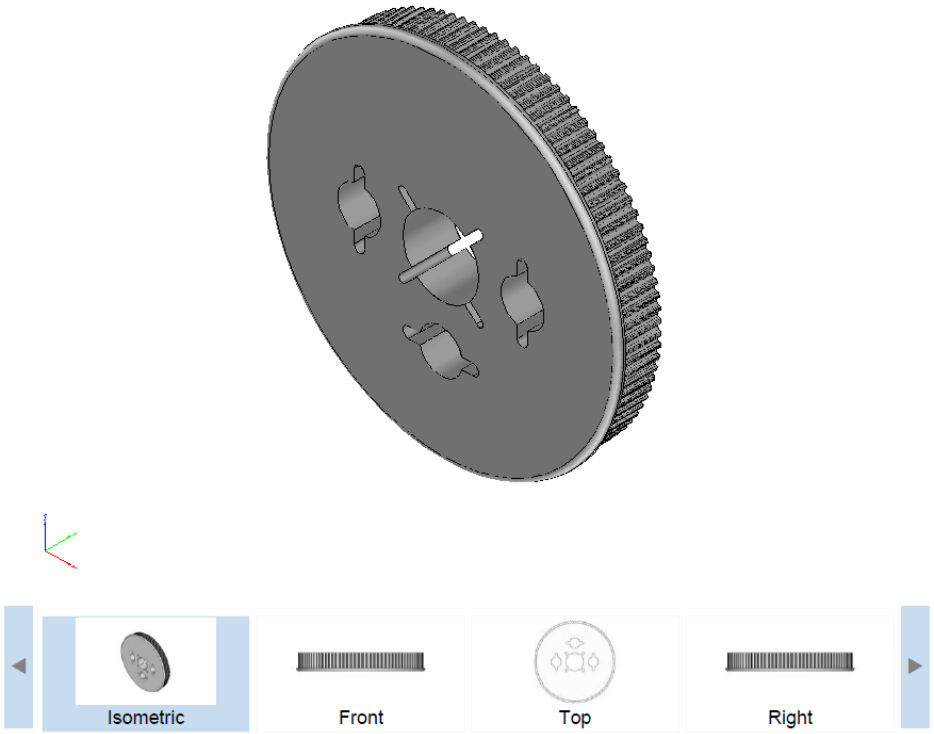
Rysunek 14: Prawy bok mocowania trzeciego silnika



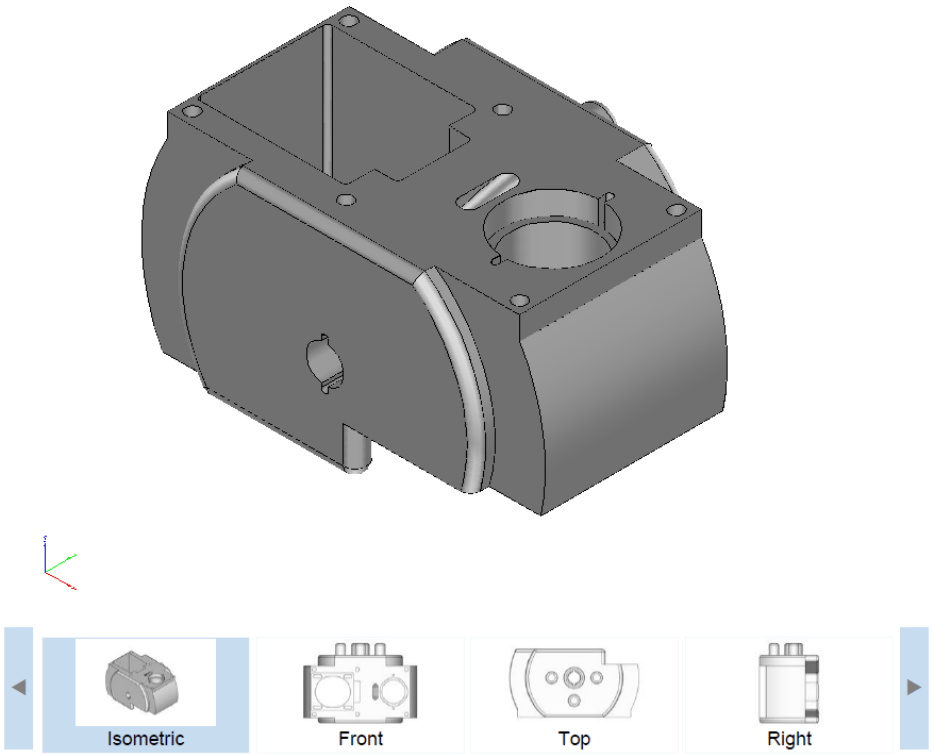
Rysunek 15: Lewy bok mocowania trzeciego silnika



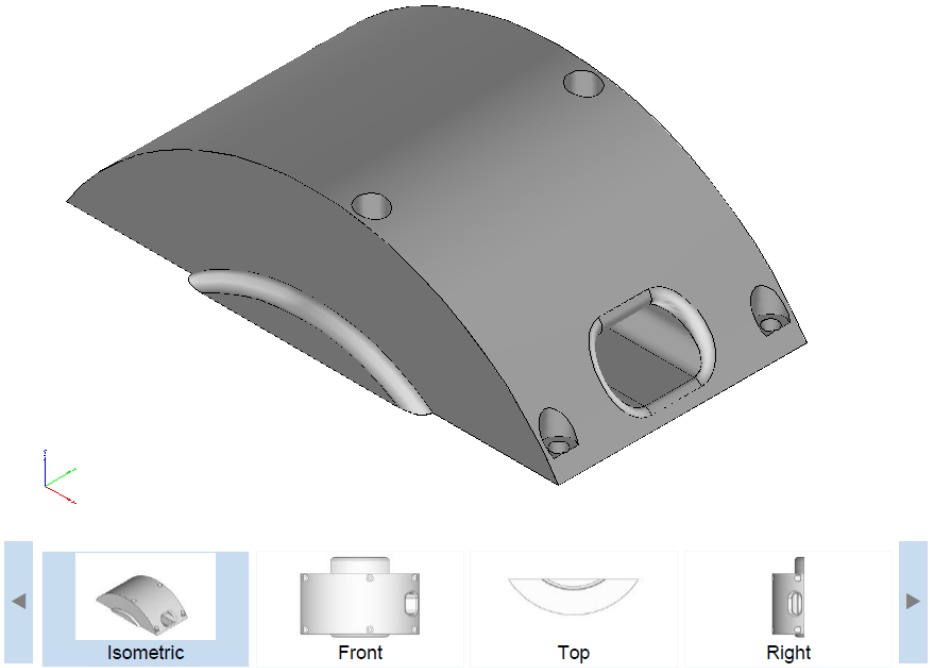
Rysunek 16: Środek mocowania trzeciego silnika



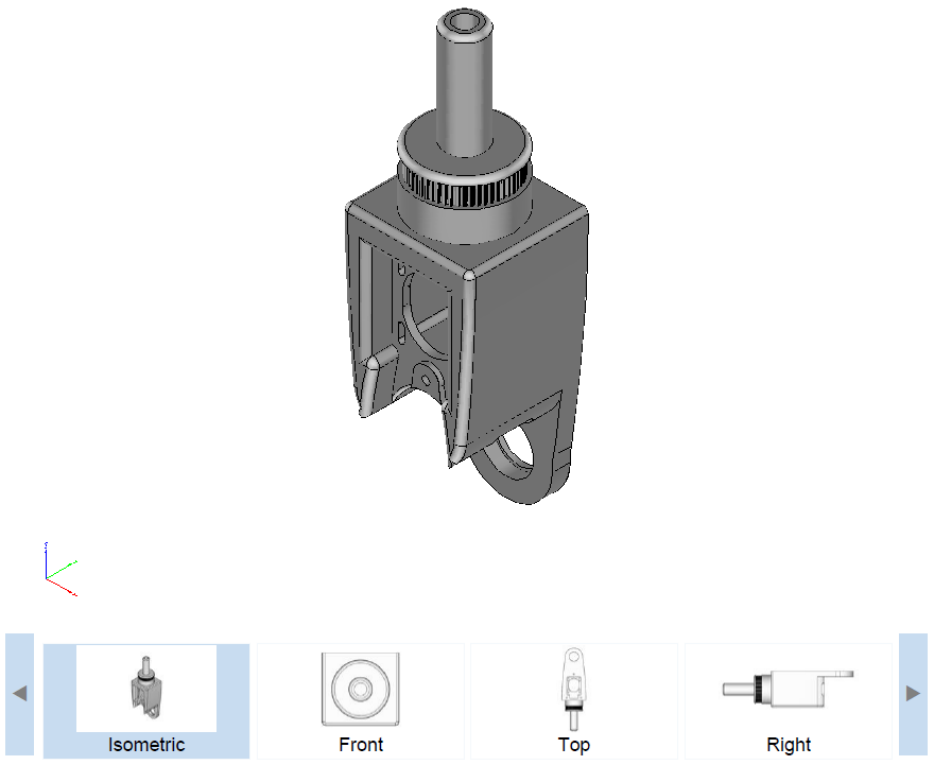
Rysunek 17: Zębatka trzeciej osi



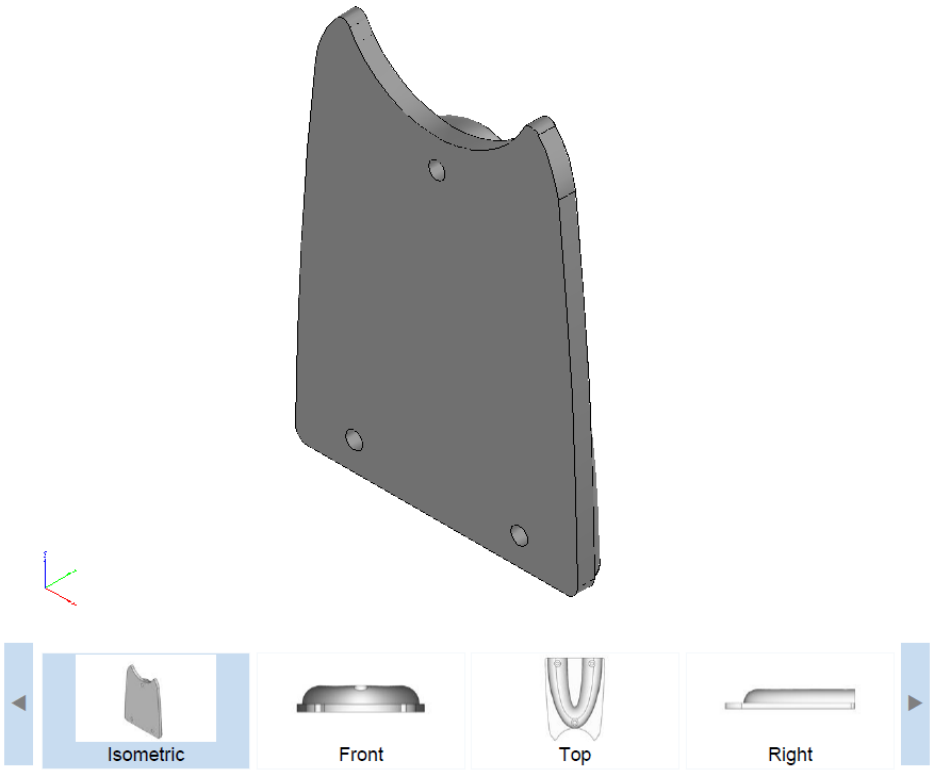
Rysunek 18: Mocowanie czwartego silnika



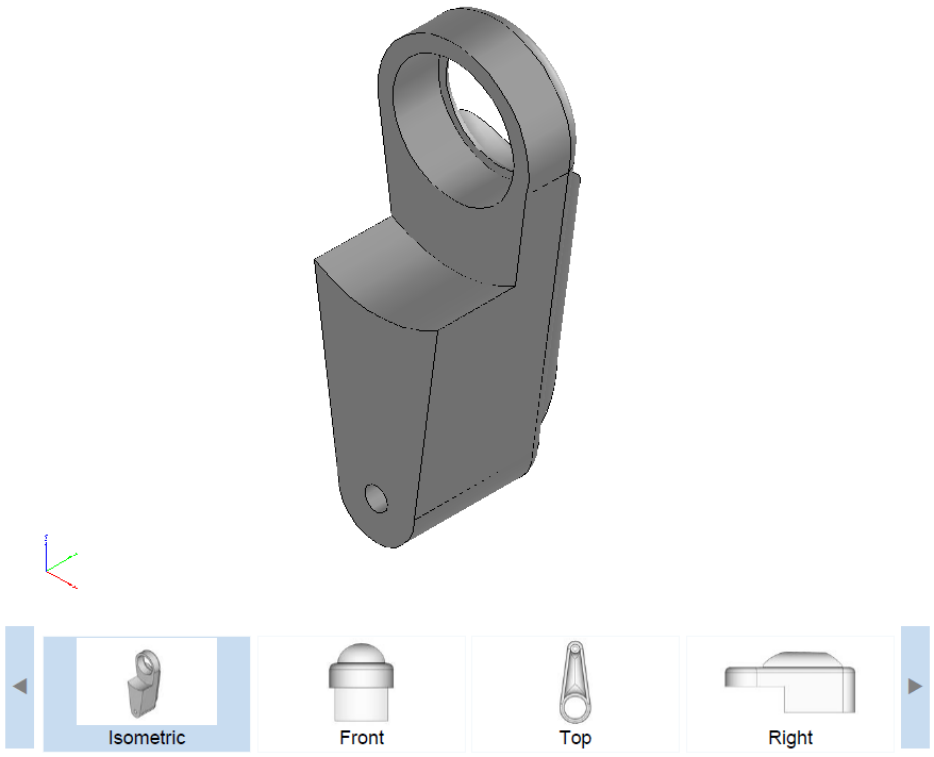
Rysunek 19: Pokrywka mocowania czwartego silnika



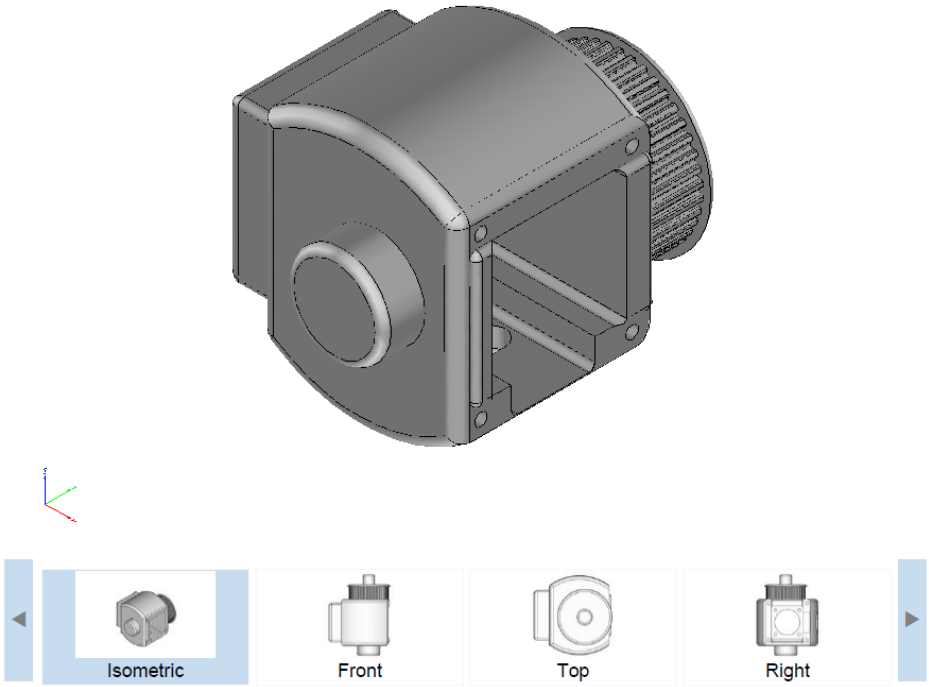
Rysunek 20: Mocowanie piątego silnika



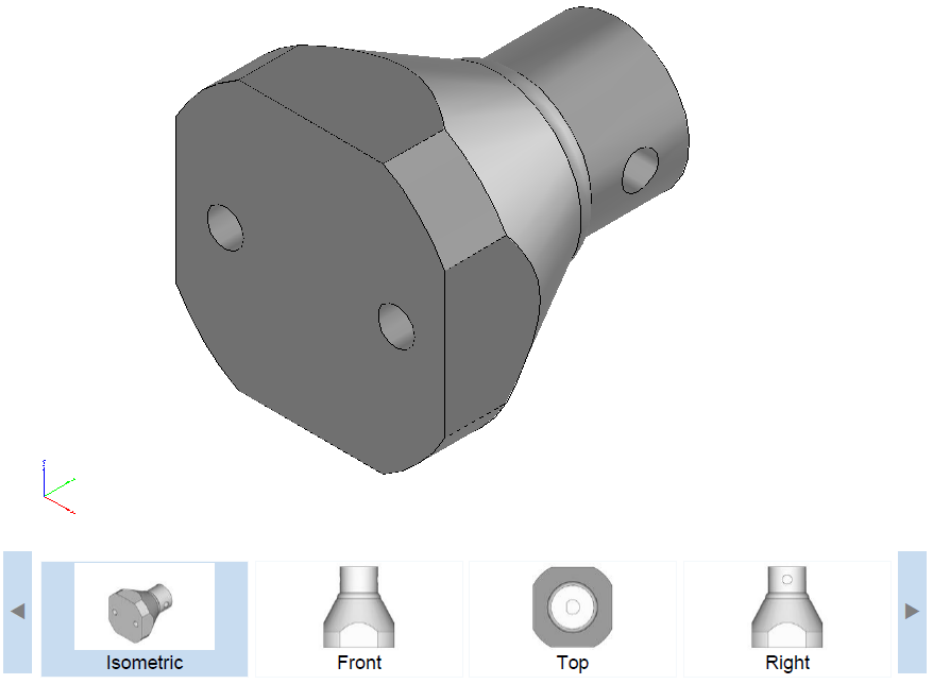
Rysunek 21: Pokrywka mocowania piątego silnika



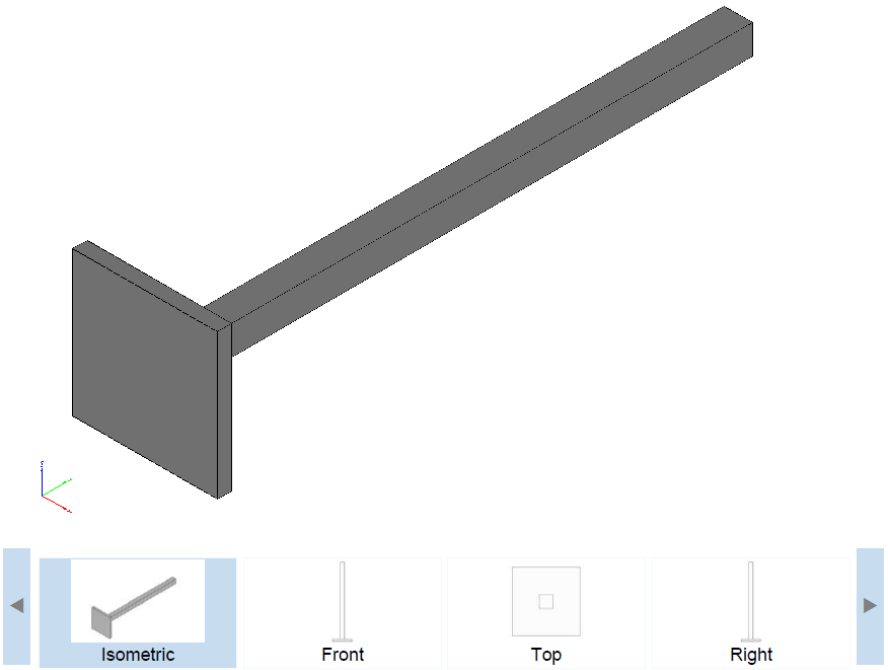
Rysunek 22: Część stabilizująca pracę piątej osi



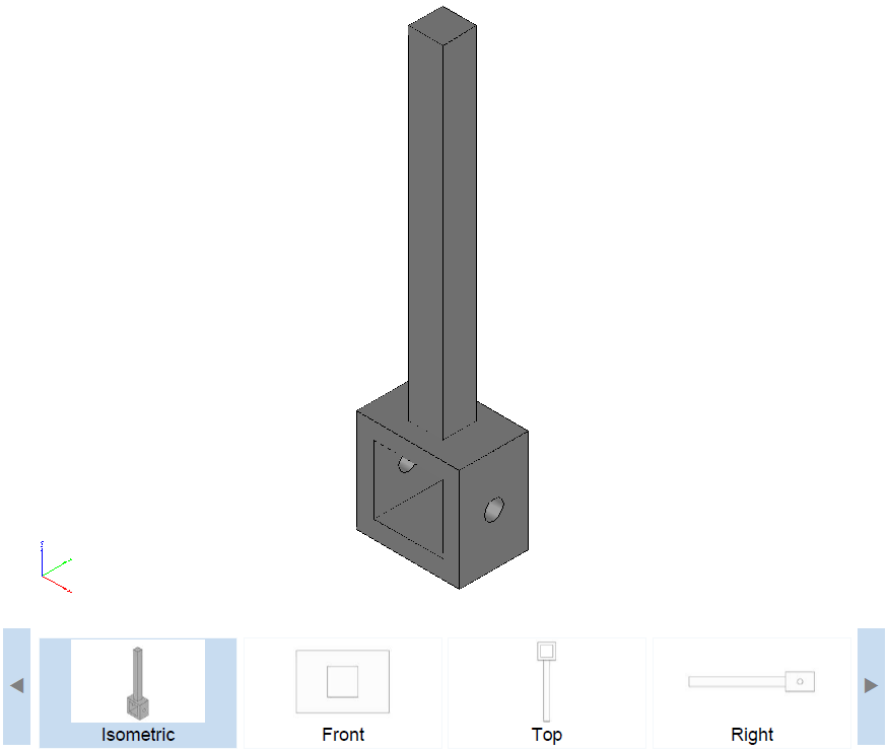
Rysunek 23: Mocowanie szóstego silnika



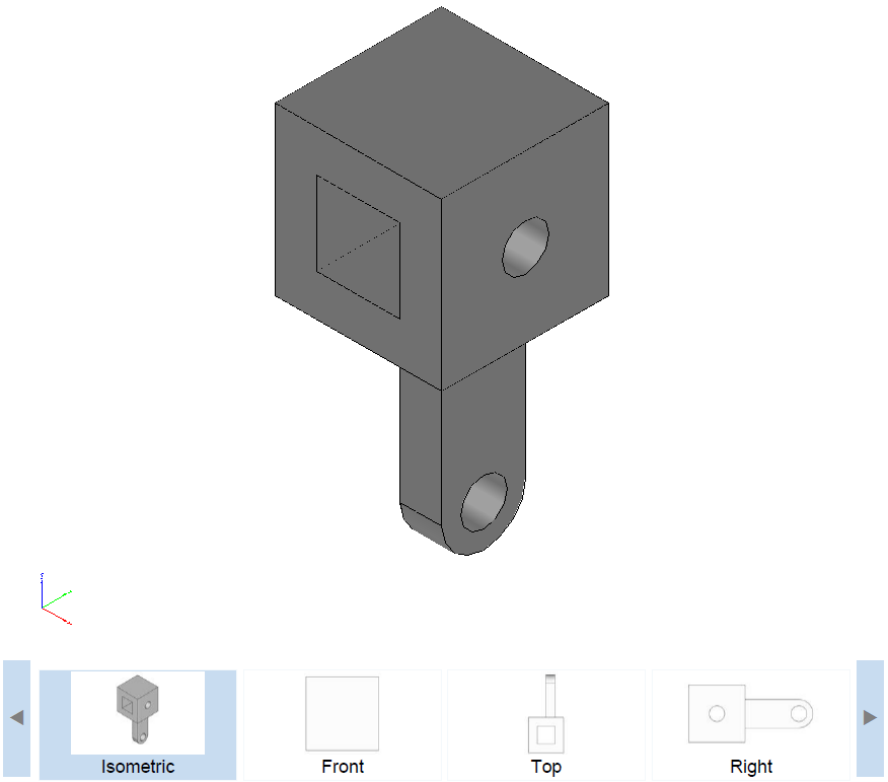
Rysunek 24: Mocowanie łapy



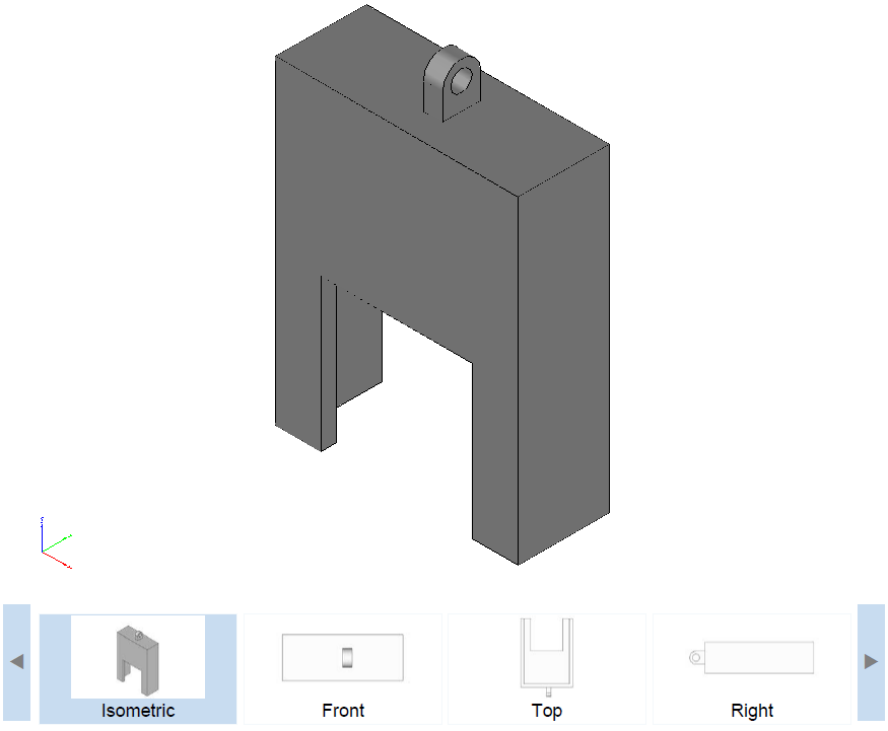
Rysunek 25: Statyw kamery - część 1



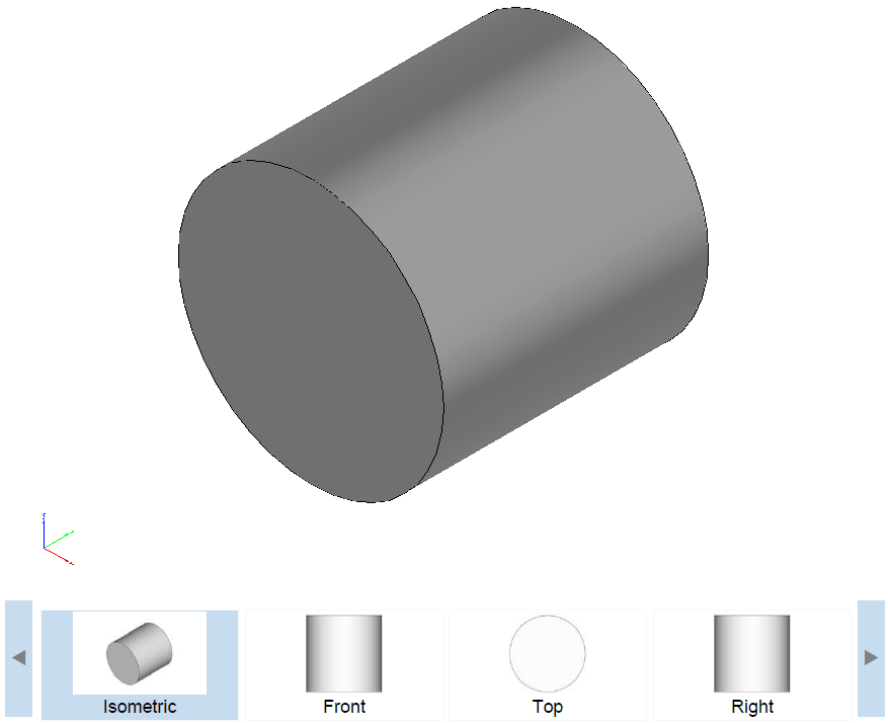
Rysunek 26: Statyw kamery - część 2



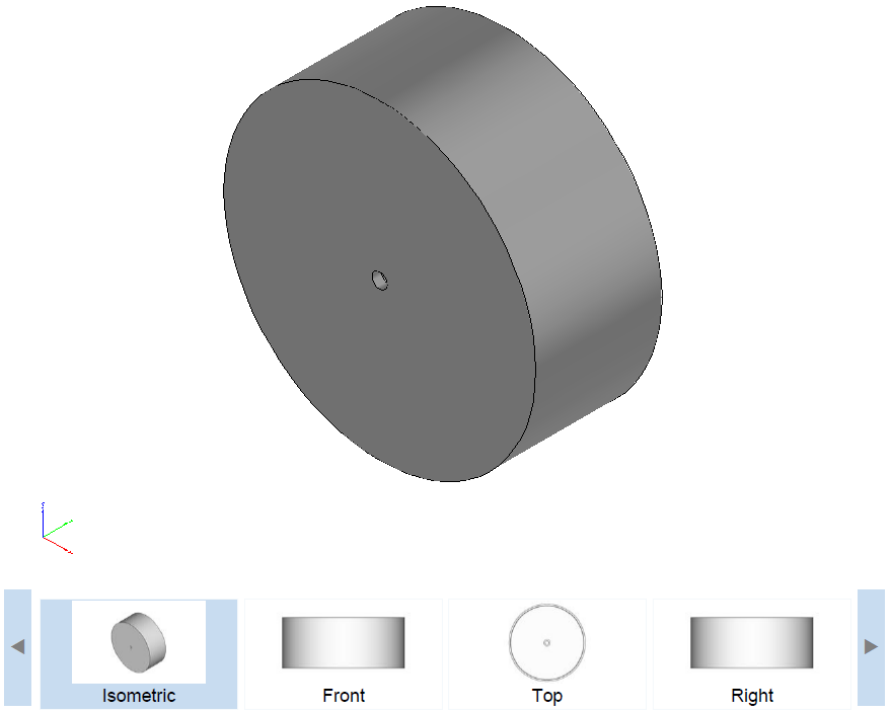
Rysunek 27: Statyw kamery - część 3



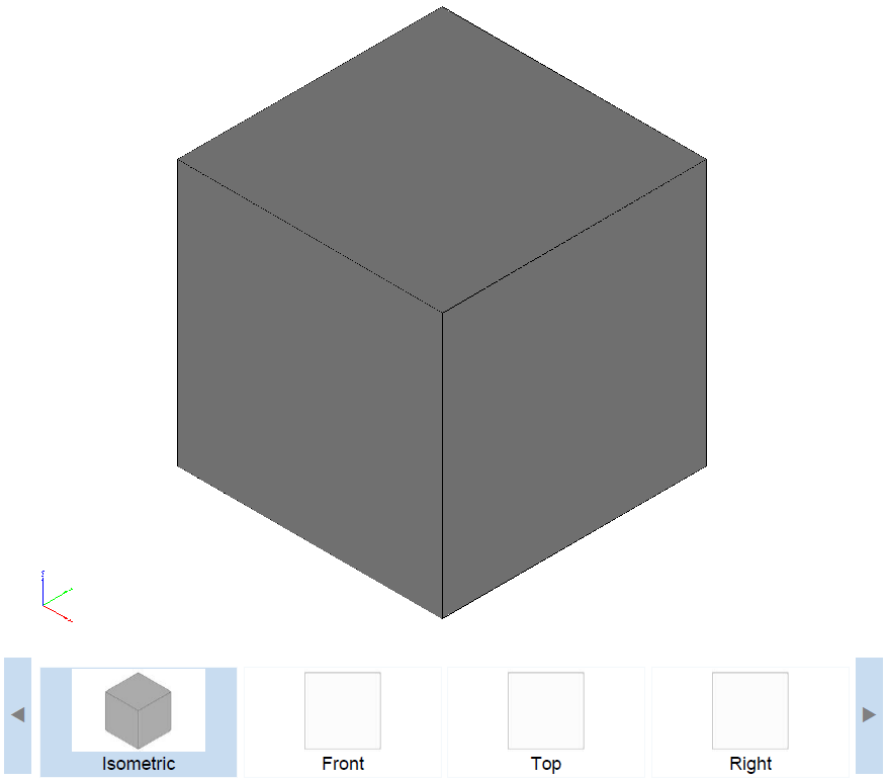
Rysunek 28: Statyw kamery - część 4



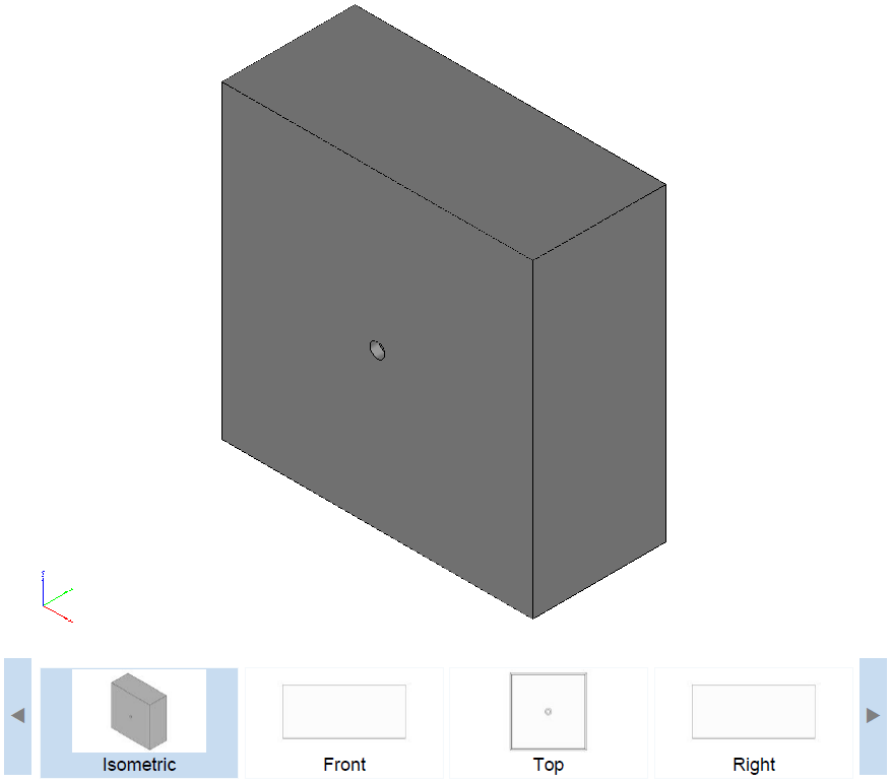
Rysunek 29: Klocek w kształcie okręgu



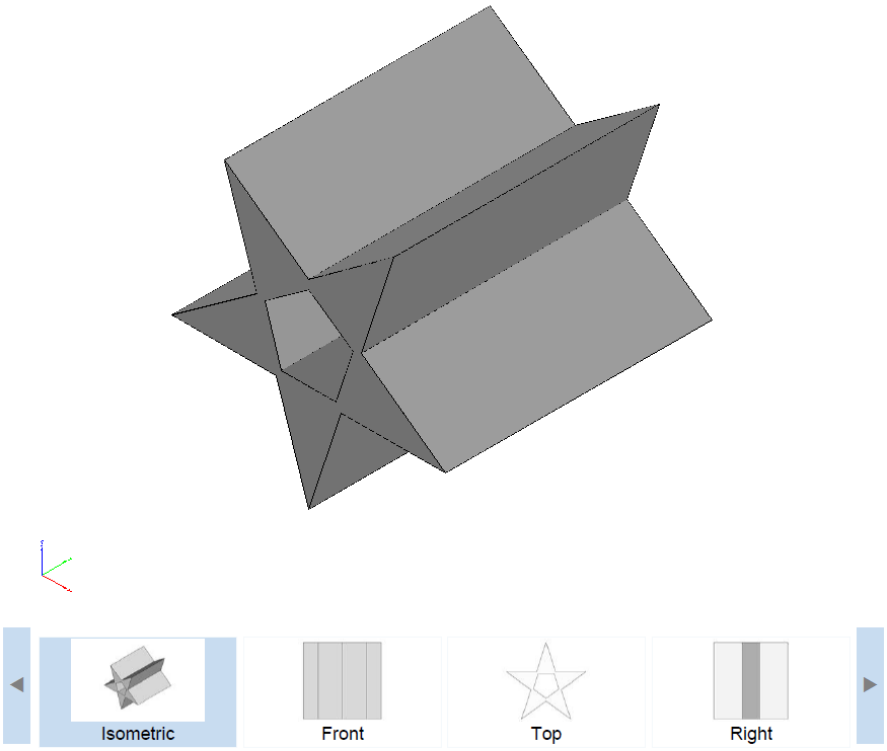
Rysunek 30: Pudełko w kształcie okręgu



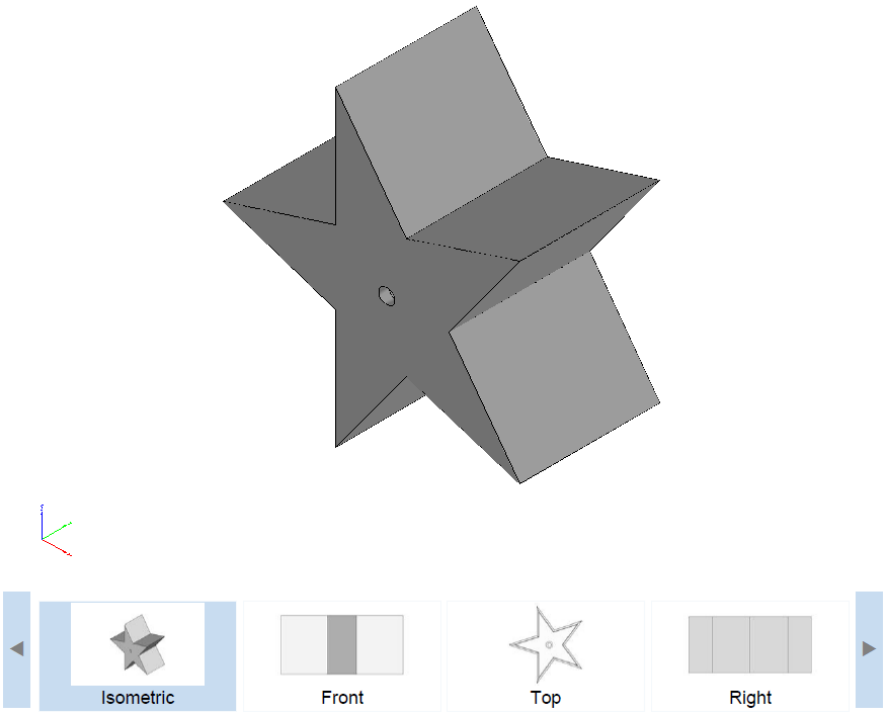
Rysunek 31: Klocek w kształcie kwadratu



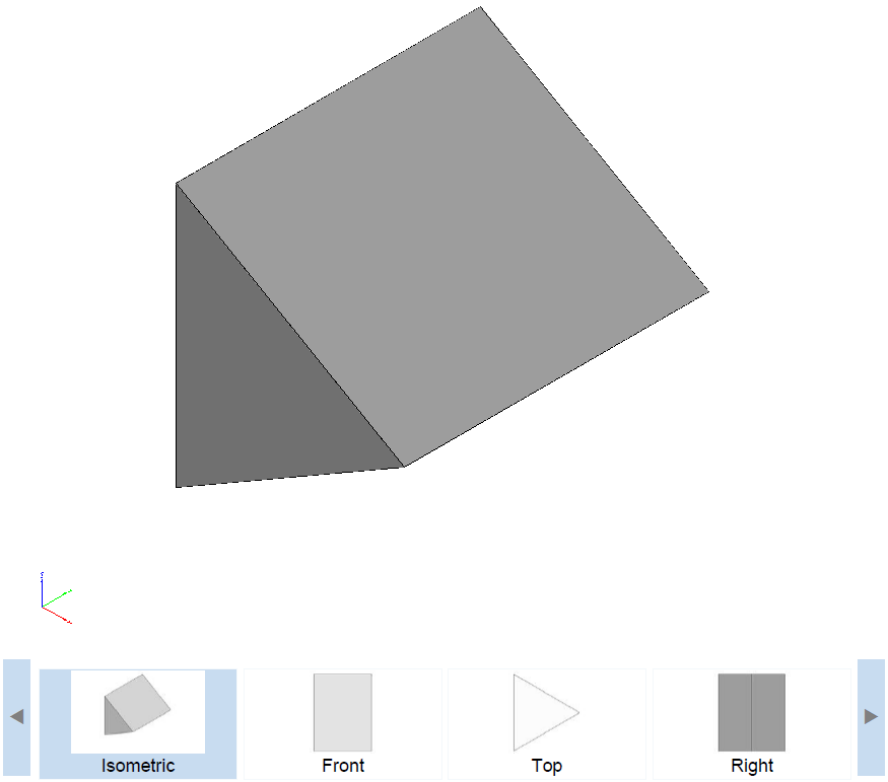
Rysunek 32: Pudełko w kształcie kwadratu



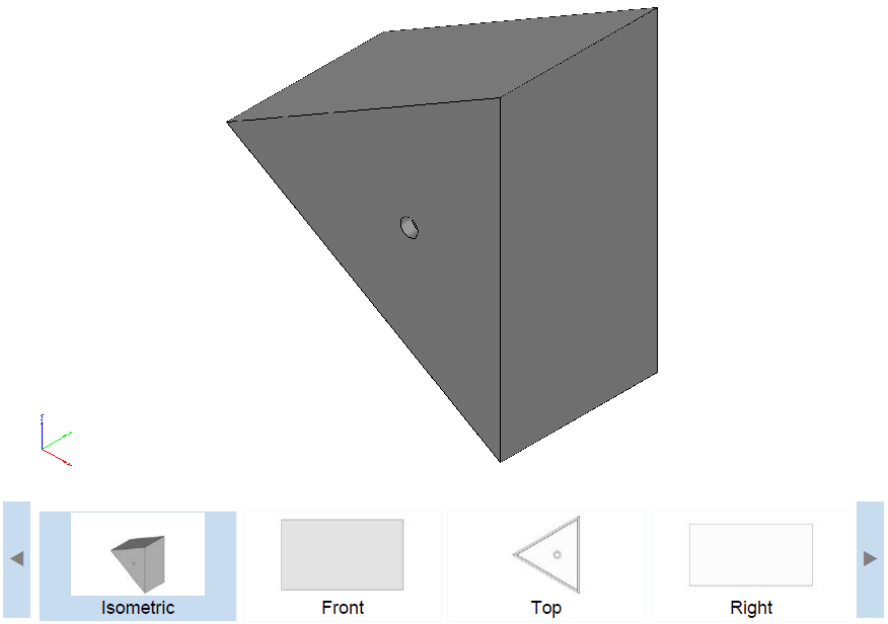
Rysunek 33: Klocek w kształcie gwiazdy



Rysunek 34: Pudełko w kształcie gwiazdy



Rysunek 35: Klocek w kształcie trójkąta



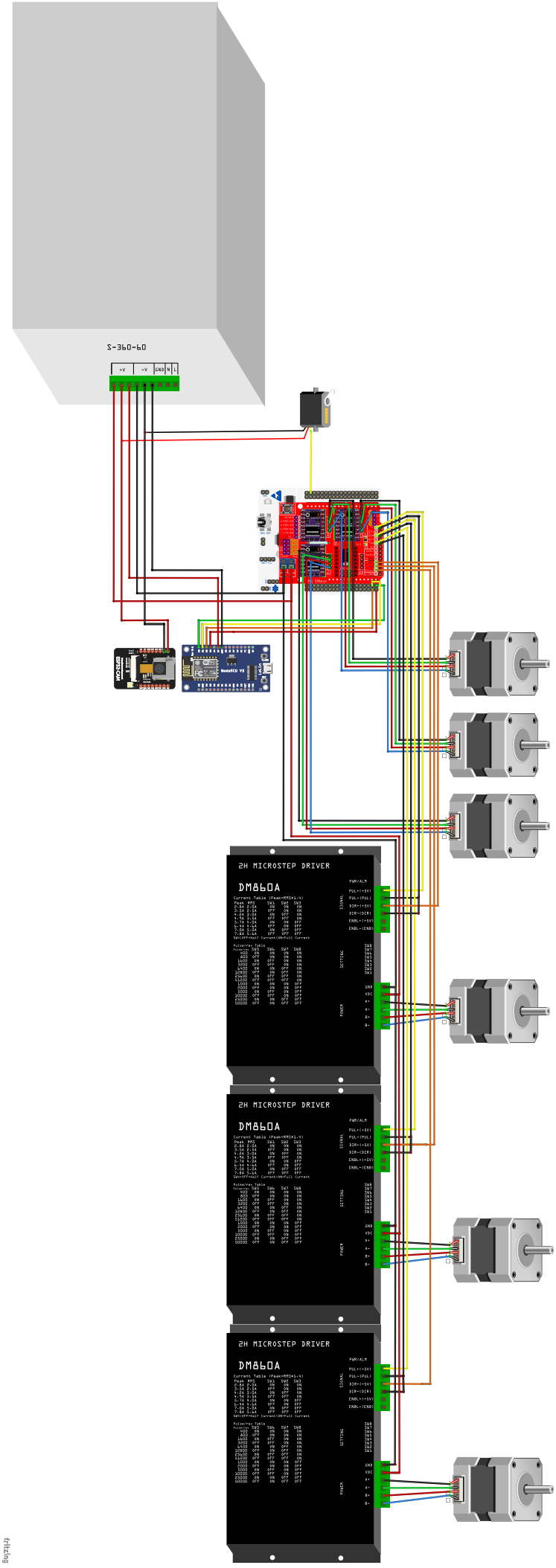
Rysunek 36: Pudełko w kształcie trójkąta

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization, Dense, Activation, Flatten
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import SparseCategoricalCrossentropy
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import os
import cv2
import math


path='C:\\Users\\magdz\\database\\test'


train= ImageDataGenerator(rescale=1/255)
validation=ImageDataGenerator(rescale=1/255)
train_dataset = train.flow_from_directory('C:\\Users\\magdz\\database\\test', target_size=(200,200), batch_size=
validation_dataset=validation.flow_from_directory('C:\\Users\\magdz\\database\\validation', target_size=(200,200
test = ImageDataGenerator(rescale=1/255)
test_dataset = test.flow_from_directory('C:\\Users\\magdz\\database\\test',target_size=(200,200))

model=Sequential(name="Robot")

#1
model.add(Conv2D(16, kernel_size=(7,7), strides=(3,3),activation='elu', padding='same', input_shape=(200,200,3))
model.add(BatchNormalization())

#2
model.add(MaxPooling2D(pool_size=(3,3), strides=(1,1)))

#3
model.add(Conv2D(32, strides=(1,1), kernel_size=(3,3), activation='elu'))
model.add(BatchNormalization())

#4
model.add(MaxPooling2D(pool_size=(2,2), strides=(1,1)))

#5
model.add(Conv2D(32, strides=(1,1), kernel_size=(3,3), activation='elu'))
model.add(BatchNormalization())

#6
model.add(MaxPooling2D(pool_size=(2,2), strides=(1,1)))

#7
model.add(Conv2D(16, strides=(1,1), kernel_size=(3,3), activation='elu'))
model.add(BatchNormalization())

#8
model.add(MaxPooling2D(pool_size=(2,2), strides=(1,1)))
model.add(Flatten())
#9
#model.add(Dense(512, activation='relu'))
model.add(Dense(9,activation='softmax'))


model.summary()
epochs=15
```

```
compute_steps_per_epoch=lambda x: int(math.ceil(1. * x / 50))
steps_per_epoch=compute_steps_per_epoch(16000)
validation_steps=compute_steps_per_epoch(8000)

model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(train_dataset,steps_per_epoch=steps_per_epoch, epochs=26,validation_data=validation_dataset,validation

model.save('C:\\Users\\magdz\\database')

result=model.evaluate(test_dataset)
print(result)
dict(zip(model.metrics_names,result))

import os
import tensorflow as tf
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
import cv2
import urllib.request
import socket



url="http://192.168.1.150/capture"
#cv2.namedWindow("robot vision",cv2.WINDOW_AUTOSIZE)
image_counter=0
while True:
    imageResponse=urllib.request.urlopen(url)
    imgnp=np.array(bytearray(imageResponse.read()),dtype=np.uint8)
    img=cv2.imdecode(imgnp,-1)
    cv2.imshow("robot_vision",img)
    k = cv2.waitKey(1)
    if k%256 ==  32:
        img_name = "frame_{}.png".format(image_counter)
        cv2.imwrite(img_name,img)
        print("screen taken")
        image_counter += 1
        break
    #key=cv2.waitKey(5)
    #if key==ord('q'):
    #    break


path='C://Users//magdz//source//repos//AI_predict//frame_0.png'

def load_image(path):
    img=image.load_img(path, target_size=(200,200))
    img_array = image.img_to_array(img)
    img_batch = np.expand_dims(img_array,axis=0)
    img_batch /= 255.0
    plt.imshow(img_batch[0])
    plt.show()
    return img_batch


#url=https://script.google.com/macros/s/AKfycbwJQX867u9EOfZBU3RKK_e_dw4vjV1e_31ugpKBbfshhyW8J1Q/exec?square=1&c
ksztalty=["kolo","kwadrat","gwiazda","trojkat"]
```

```
model=tf.keras.models.load_model('C://Users//magdz//robotAI')
pred=model.predict(load_image(path))
print(pred)
pred_val=np.argmax(pred)
print("ksztalt:",ksztalty[np.argmax(pred)])
s=socket.socket()
port=80
ip="192.168.1.150"
s.connect((ip,port))
if pred_val == 0:#dopasowane do kolejnosci na stm 0-kwadrat 1-kolo 2-trojkat 3-gwiazda
    message="1"
    s.send(message.encode())
    print("wyslano")
elif pred_val == 1:
    message="0"
    s.send(message.encode())
    print("wyslano")
elif pred_val == 2:
    message="3"
    s.send(message.encode())
    print("wyslano")
elif pred_val == 3:
    message="2"
    s.send(message.encode())
    print("wyslano")
```


S-360-60
+V
-V
GND
N
L
2H MICROSTEP DRIVER
DM860A
2H MICROSTEP DRIVER
DM860A
2H MICROSTEP DRIVER
DM860A

```c
/**
  ******************************************************************************
  * File Name          : gpio.c
  * Description        : This file provides code for the configuration
  *                      of all used GPIO pins.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "gpio.h"
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/*----------------------------------------------------------------------------*/
/* Configure GPIO                                                             */
/*----------------------------------------------------------------------------*/
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
*/
void MX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, DIR_AXIS3_Pin|DIR_AXIS2_Pin|DIR_AXIS1_Pin|GPIO_PIN_5
                          |DIR_AXIS6_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, DIR_AXIS5_Pin|DIR_AXIS4_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : PCPin PCPin PCPin PCPin */
  GPIO_InitStruct.Pin =
SensorSquare_Pin|SensorRoller_Pin|SensorTriangle_Pin|SensorStar_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
```

```
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : PAPin PAPin PAPin PA5
                           PAPin */
  GPIO_InitStruct.Pin = DIR_AXIS3_Pin|DIR_AXIS2_Pin|DIR_AXIS1_Pin|GPIO_PIN_5
                        |DIR_AXIS6_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PBPin PBPin */
  GPIO_InitStruct.Pin = DIR_AXIS5_Pin|DIR_AXIS4_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI0_IRQn);

  HAL_NVIC_SetPriority(EXTI1_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI1_IRQn);

  HAL_NVIC_SetPriority(EXTI2_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI2_IRQn);

  HAL_NVIC_SetPriority(EXTI3_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI3_IRQn);

}

/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/

```

```c
/**
  ******************************************************************************
  * File Name          : gpio.h
  * Description        : This file contains all the functions prototypes for
  *                      the gpio
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __gpio_H
#define __gpio_H
#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

void MX_GPIO_Init(void);

/* USER CODE BEGIN Prototypes */

/* USER CODE END Prototypes */

#ifdef __cplusplus
}
#endif
#endif /*__ pinoutConfig_H */

/**
  * @}
  */

/**
  * @}
  */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/

```

```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stepmotor1.h"
#include "stepmotor2.h"
#include "stepmotor3.h"
#include "stepmotor4.h"
#include "stepmotor5.h"
#include "stepmotor6.h"
#include "servomotor7.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
volatile uint16_t figure = 0;

volatile uint16_t stepCounter1 = 0;
volatile uint16_t stepLimit1 = 0;
volatile uint16_t isStop1 = 1;
volatile int rotationCounter1 = 0;

volatile uint16_t stepCounter2 = 0;
volatile uint16_t stepLimit2 = 0;
```

```
63 volatile uint16_t isStop2 = 1;
64 volatile int rotationCounter2 = 0;
65
66 volatile uint16_t stepCounter3 = 0;
67 volatile uint16_t stepLimit3 = 0;
68 volatile uint16_t isStop3 = 1;
69 volatile int rotationCounter3 = 0;
70
71 volatile uint16_t stepCounter4 = 0;
72 volatile uint16_t stepLimit4 = 0;
73 volatile uint16_t isStop4 = 1;
74 volatile int rotationCounter4 = 0;
75
76 volatile uint16_t stepCounter5 = 0;
77 volatile uint16_t stepLimit5 = 0;
78 volatile uint16_t isStop5 = 1;
79 volatile int rotationCounter5 = 0;
80
81 volatile uint16_t stepCounter6 = 0;
82 volatile uint16_t stepLimit6 = 0;
83 volatile uint16_t isStop6 = 1;
84 volatile int rotationCounter6 = 0;
85 /* USER CODE END PV */
86
87 /* Private function prototypes -----------------------------------------------*/
88 void SystemClock_Config(void);
89 /* USER CODE BEGIN PFP */
90 void pick_move(void);
91 void circle_move(void);
92 void square_move(void);
93 void home_move(void);
94 void star_move(void);
95 void triangle_move(void);
96 /* USER CODE END PFP */
97
98 /* Private user code ---------------------------------------------------------*/
99 /* USER CODE BEGIN 0 */
100 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
101 {
102     if(htim == &htim8)
103     {
104         stepCounter1 = stepCounter1 + 1;
105         if(stepCounter1 >= stepLimit1)
106         {
107             HAL_TIM_PWM_Stop(&htim8, TIM_CHANNEL_2);
108             stepCounter1 = 0;
109             isStop1 = 1;
110         }
111     }
112     if(htim == &htim4)
113     {
114         stepCounter2 = stepCounter2 + 1;
115         if(stepCounter2 >= stepLimit2)
116         {
117             HAL_TIM_PWM_Stop(&htim4, TIM_CHANNEL_1);
118             stepCounter2 = 0;
119             isStop2 = 1;
120         }
121     }
122     if(htim == &htim14)
123     {
124         stepCounter3 = stepCounter3 + 1;
```

```
125         if(stepCounter3 >= stepLimit3)
126         {
127             HAL_TIM_PWM_Stop(&htim14, TIM_CHANNEL_1);
128             stepCounter3 = 0;
129             isStop3 = 1;
130         }
131     }
132 
133     if(htim == &htim1)
134     {
135         stepCounter4 = stepCounter4 + 1;
136         if(stepCounter4 >= stepLimit4)
137         {
138             HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_3);
139             stepCounter4 = 0;
140             isStop4 = 1;
141         }
142     }
143     if(htim == &htim2)
144     {
145         stepCounter5 = stepCounter5 + 1;
146         if(stepCounter5 >= stepLimit5)
147         {
148             HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_2);
149             stepCounter5 = 0;
150             isStop5 = 1;
151         }
152     }
153     if(htim == &htim3)
154     {
155         stepCounter6 = stepCounter6 + 1;
156         if(stepCounter6 >= stepLimit6)
157         {
158             HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_2);
159             stepCounter6 = 0;
160             isStop6 = 1;
161         }
162     }
163 }
164 /* USER CODE END 0 */
165 
166 /**
167   * @brief  The application entry point.
168   * @retval int
169   */
170 int main(void)
171 {
172   /* USER CODE BEGIN 1 */
173 
174   /* USER CODE END 1 */
175 
176   /* MCU Configuration--------------------------------------------------------*/
177 
178   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
179   HAL_Init();
180 
181   /* USER CODE BEGIN Init */
182 
183   /* USER CODE END Init */
184 
185   /* Configure the system clock */
186   SystemClock_Config();
```

```
187
188  /* USER CODE BEGIN SysInit */
189
190  /* USER CODE END SysInit */
191
192  /* Initialize all configured peripherals */
193  MX_GPIO_Init();
194  MX_USART2_UART_Init();
195  MX_TIM1_Init();
196  MX_TIM2_Init();
197  MX_TIM3_Init();
198  MX_TIM4_Init();
199  MX_TIM8_Init();
200  MX_TIM14_Init();
201  MX_TIM10_Init();
202  /* USER CODE BEGIN 2 */
203  ////////////////////
204  stepper_init_motor1();
205  stepper_init_motor2();
206  stepper_init_motor3();
207  stepper_init_motor4();
208  stepper_init_motor5();
209  stepper_init_motor6();
210  servo_init_motor7();
211  ////////////////////
212  //HAL_TIM_Base_Start_IT(&htim1);
213  //HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
214  /* USER CODE END 2 */
215
216  /* Infinite loop */
217  /* USER CODE BEGIN WHILE */
218  HAL_Delay(500);
219  servo_close_gripper();
220  HAL_Delay(3000);
221  while (1)
222  {
223      if(figure == 1)
224      {
225        pick_move();
226        square_move();
227        home_move();
228      }
229      if(figure == 2)
230      {
231        pick_move();
232        circle_move();
233        home_move();
234      }
235      if(figure == 3)
236      {
237        pick_move();
238        triangle_move();
239        home_move();
240      }
241      if(figure == 4)
242      {
243        pick_move();
244        star_move();
245        home_move();
246      }
247
248
```

```
249
250     /* USER CODE END WHILE */
251
252     /* USER CODE BEGIN 3 */
253   }
254   /* USER CODE END 3 */
255 }
256
257 /**
258   * @brief System Clock Configuration
259   * @retval None
260   */
261 void SystemClock_Config(void)
262 {
263   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
264   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
265
266   /** Configure the main internal regulator output voltage
267   */
268   __HAL_RCC_PWR_CLK_ENABLE();
269   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);
270   /** Initializes the RCC Oscillators according to the specified parameters
271   * in the RCC_OscInitTypeDef structure.
272   */
273   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
274   RCC_OscInitStruct.HSIState = RCC_HSI_ON;
275   RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
276   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
277   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
278   RCC_OscInitStruct.PLL.PLLM = 8;
279   RCC_OscInitStruct.PLL.PLLN = 84;
280   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
281   RCC_OscInitStruct.PLL.PLLQ = 2;
282   RCC_OscInitStruct.PLL.PLLR = 2;
283   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
284   {
285     Error_Handler();
286   }
287   /** Initializes the CPU, AHB and APB buses clocks
288   */
289   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
290                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
291   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
292   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
293   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
294   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
295
296   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
297   {
298     Error_Handler();
299   }
300 }
301
302 /* USER CODE BEGIN 4 */
303 /* Ruch robota w kierunku kwadratu */
304 void pick_move(void)
305 {
306     figure = 0;
307     HAL_Delay(2000);
308     servo_open_gripper();
309     stepper_rot_motor2(4000, 5, 0, &stepLimit2, &rotationCounter2, &isStop2);
310     stepper_rot_motor3(3500, 5, 0, &stepLimit3, &rotationCounter3, &isStop3);
```

```
311     stepper_rot_motor6(900, 1, 1, &stepLimit6, &rotationCounter6, &isStop6);
312     HAL_Delay(3000);
313     stepper_rot_motor5(250, 1, 1, &stepLimit5, &rotationCounter5, &isStop5);
314     while(isStop2 == 0);
315     while(isStop3 == 0);
316     while(isStop5 == 0);
317     while(isStop6 == 0);
318     HAL_Delay(500);
319     servo_close_gripper();
320 }
321
322 void square_move(void)
323 {
324     stepper_rot_motor2(2000, 5, 1, &stepLimit2, &rotationCounter2, &isStop2);
325     stepper_rot_motor3(1000, 5, 1, &stepLimit3, &rotationCounter3, &isStop3);
326     stepper_rot_motor5(125, 1, 0, &stepLimit5, &rotationCounter5, &isStop5);
327     while(isStop2 == 0);
328     while(isStop3 == 0);
329     while(isStop5 == 0);
330     stepper_rot_motor2(100, 5, 0, &stepLimit2, &rotationCounter2, &isStop2);
331     stepper_rot_motor5(80, 1, 1, &stepLimit5, &rotationCounter5, &isStop5);
332     while(isStop2 == 0);
333     stepper_rot_motor2(1500, 4, 0, &stepLimit2, &rotationCounter2, &isStop2);
334     stepper_rot_motor1(1350, 5, 0, &stepLimit1, &rotationCounter1, &isStop1);
335     stepper_rot_motor3(2000, 5, 0, &stepLimit3, &rotationCounter3, &isStop3);
336     while(isStop2 == 0);
337     while(isStop3 == 0);
338     while(isStop5 == 0);
339     while(isStop1 == 0);
340     HAL_Delay(500);
341     servo_open_gripper();
342 }
343
344 void circle_move(void)
345 {
346     stepper_rot_motor2(2000, 5, 1, &stepLimit2, &rotationCounter2, &isStop2);
347     stepper_rot_motor3(1000, 5, 1, &stepLimit3, &rotationCounter3, &isStop3);
348     stepper_rot_motor5(125, 1, 0, &stepLimit5, &rotationCounter5, &isStop5);
349     while(isStop2 == 0);
350     while(isStop3 == 0);
351     while(isStop5 == 0);
352     stepper_rot_motor2(100, 5, 0, &stepLimit2, &rotationCounter2, &isStop2);
353     stepper_rot_motor5(80, 1, 1, &stepLimit5, &rotationCounter5, &isStop5);
354     while(isStop2 == 0);
355     stepper_rot_motor2(1500, 4, 0, &stepLimit2, &rotationCounter2, &isStop2);
356     stepper_rot_motor1(1350, 5, 1, &stepLimit1, &rotationCounter1, &isStop1);
357     stepper_rot_motor3(2000, 5, 0, &stepLimit3, &rotationCounter3, &isStop3);
358     while(isStop2 == 0);
359     while(isStop3 == 0);
360     while(isStop5 == 0);
361     while(isStop1 == 0);
362     HAL_Delay(500);
363     servo_open_gripper();
364 }
365
366 void triangle_move(void)
367 {
368     stepper_rot_motor2(2500, 5, 1, &stepLimit2, &rotationCounter2, &isStop2);
369     stepper_rot_motor3(1500, 5, 1, &stepLimit3, &rotationCounter3, &isStop3);
370     stepper_rot_motor5(125, 1, 0, &stepLimit5, &rotationCounter5, &isStop5);
371     while(isStop2 == 0);
372     while(isStop3 == 0);
```

```
373     while(isStop5 == 0);
374     stepper_rot_motor2(100, 5, 0, &stepLimit2, &rotationCounter2, &isStop2);
375     stepper_rot_motor5(100, 1, 0, &stepLimit5, &rotationCounter5, &isStop5);
376     while(isStop2 == 0);
377     stepper_rot_motor1(4050, 6, 1, &stepLimit1, &rotationCounter1, &isStop1);
378     while(isStop5 == 0);
379     while(isStop1 == 0);
380     HAL_Delay(500);
381     servo_open_gripper();
382 }
383 
384 void star_move(void)
385 {
386     stepper_rot_motor2(2400, 5, 1, &stepLimit2, &rotationCounter2, &isStop2);
387     stepper_rot_motor3(1400, 5, 1, &stepLimit3, &rotationCounter3, &isStop3);
388     stepper_rot_motor5(125, 1, 0, &stepLimit5, &rotationCounter5, &isStop5);
389     while(isStop2 == 0);
390     while(isStop3 == 0);
391     while(isStop5 == 0);
392     stepper_rot_motor2(100, 5, 0, &stepLimit2, &rotationCounter2, &isStop2);
393     stepper_rot_motor5(100, 1, 0, &stepLimit5, &rotationCounter5, &isStop5);
394     while(isStop2 == 0);
395     stepper_rot_motor1(4050, 6, 0, &stepLimit1, &rotationCounter1, &isStop1);
396     while(isStop5 == 0);
397     while(isStop1 == 0);
398     HAL_Delay(500);
399     servo_open_gripper();
400 }
401 
402 void home_move(void)
403 {
404     HAL_Delay(500);
405     stepper_rot_home_motor2(5, 0, &stepLimit2, &rotationCounter2, &isStop2);
406     stepper_rot_home_motor3(5, 0, &stepLimit3, &rotationCounter3, &isStop3);
407     HAL_Delay(1000);
408     stepper_rot_home_motor1(5, 0, &stepLimit1, &rotationCounter1, &isStop1);
409     stepper_rot_home_motor5(1, 0, &stepLimit5, &rotationCounter5, &isStop5);
410     stepper_rot_home_motor6(1, 0, &stepLimit6, &rotationCounter6, &isStop6);
411     while(isStop1 == 0);
412     while(isStop2 == 0);
413     while(isStop3 == 0);
414     while(isStop5 == 0);
415     while(isStop6 == 0);
416     servo_close_gripper();
417     HAL_Delay(500);
418 }
419 
420 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
421 {
422     if(GPIO_Pin == SensorSquare_Pin)
423     {
424         figure = 1;
425     }
426     if(GPIO_Pin == SensorRoller_Pin)
427     {
428         figure = 2;
429     }
430     if(GPIO_Pin == SensorTriangle_Pin)
431     {
432         figure = 3;
433     }
434     if(GPIO_Pin == SensorStar_Pin)
```

```
435     {
436         figure = 4;
437     }
438 }
439 
440 /* USER CODE END 4 */
441 
442 /**
443   * @brief  This function is executed in case of error occurrence.
444   * @retval None
445   */
446 void Error_Handler(void)
447 {
448   /* USER CODE BEGIN Error_Handler_Debug */
449   /* User can add his own implementation to report the HAL error return state */
450 
451   /* USER CODE END Error_Handler_Debug */
452 }
453 
454 #ifdef  USE_FULL_ASSERT
455 /**
456   * @brief  Reports the name of the source file and the source line number
457   *         where the assert_param error has occurred.
458   * @param  file: pointer to the source file name
459   * @param  line: assert_param error line source number
460   * @retval None
461   */
462 void assert_failed(uint8_t *file, uint32_t line)
463 {
464   /* USER CODE BEGIN 6 */
465   /* User can add his own implementation to report the file name and line number,
466      tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
467   /* USER CODE END 6 */
468 }
469 #endif /* USE_FULL_ASSERT */
470 
471 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
472 
```

```c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/
#define SensorSquare_Pin GPIO_PIN_0
#define SensorSquare_GPIO_Port GPIOC
```

```
63 #define SensorSquare_EXTI_IRQn EXTI0_IRQn
64 #define SensorRoller_Pin GPIO_PIN_1
65 #define SensorRoller_GPIO_Port GPIOC
66 #define SensorRoller_EXTI_IRQn EXTI1_IRQn
67 #define SensorTriangle_Pin GPIO_PIN_2
68 #define SensorTriangle_GPIO_Port GPIOC
69 #define SensorTriangle_EXTI_IRQn EXTI2_IRQn
70 #define SensorStar_Pin GPIO_PIN_3
71 #define SensorStar_GPIO_Port GPIOC
72 #define SensorStar_EXTI_IRQn EXTI3_IRQn
73 #define DIR_AXIS3_Pin GPIO_PIN_0
74 #define DIR_AXIS3_GPIO_Port GPIOA
75 #define DIR_AXIS2_Pin GPIO_PIN_1
76 #define DIR_AXIS2_GPIO_Port GPIOA
77 #define DIR_AXIS1_Pin GPIO_PIN_4
78 #define DIR_AXIS1_GPIO_Port GPIOA
79 #define DIR_AXIS5_Pin GPIO_PIN_10
80 #define DIR_AXIS5_GPIO_Port GPIOB
81 #define DIR_AXIS6_Pin GPIO_PIN_8
82 #define DIR_AXIS6_GPIO_Port GPIOA
83 #define DIR_AXIS4_Pin GPIO_PIN_4
84 #define DIR_AXIS4_GPIO_Port GPIOB
85 /* USER CODE BEGIN Private defines */
86
87 /* USER CODE END Private defines */
88
89 #ifdef __cplusplus
90 }
91 #endif
92
93 #endif /* __MAIN_H */
94
95 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
96
```

```c
#include "stepmotor1.h"
#include "tim.h"
#include "gpio.h"
#include "main.h"


/* inicjalizacja biblioteki */
void stepper_init_motor1(void)
{
    HAL_TIM_Base_Start_IT(&PULSE_TIM_MOTOR1);
    HAL_Delay(2);
}

/* ustawia predkosc obrotowa silnika krokowego w rpm */
void stepper_speed_motor1(uint16_t rpm)
{
    if(rpm > 0)
    {
        uint16_t arr_val = XT_TIM_CLK_MOTOR1 / (((XT_TIM_PSC_MOTOR1+1) * rpm * STEPS_PER_REV_MOTOR1 * MICROSTEP_NUM_MOTOR1) / 60) - 1;
        uint16_t pulse_val = arr_val / 2;
        __HAL_TIM_SET_AUTORELOAD(&PULSE_TIM_MOTOR1, arr_val);
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR1, PULSE_TIM_CH_MOTOR1, pulse_val);
    }
    else
    {
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR1, PULSE_TIM_CH_MOTOR1, 0);
    }
}

/* rozpoczyna ruch silnika z kierunkiem obrotu dir */
void stepper_run_motor1(uint8_t dir)
{
    if(dir == STEPPER_CW_MOTOR1)
    {
        HAL_GPIO_WritePin(DIR_AXIS1_GPIO_Port, DIR_AXIS1_Pin, GPIO_PIN_SET);
    }
    else if (dir == STEPPER_CCW_MOTOR1)
    {
        HAL_GPIO_WritePin(DIR_AXIS1_GPIO_Port, DIR_AXIS1_Pin, GPIO_PIN_RESET);
    }
    HAL_Delay(2);
    HAL_TIM_PWM_Start(&PULSE_TIM_MOTOR1, PULSE_TIM_CH_MOTOR1);
}

/* zatrzymuje silnik - wylacza wyjscia kontrolera silnika */
void stepper_stop_motor1(void)
{
    HAL_TIM_PWM_Stop(&PULSE_TIM_MOTOR1, PULSE_TIM_CH_MOTOR1);
}

/* ustawia zadana liczbe krokow do wykonania */
void stepper_steps_motor1(uint16_t steps, volatile uint16_t *stepLimit1)
{
    *stepLimit1 = steps;
}

/* obraca wal silnika o zadny kat z zadana predkoscia w kierunku dir */
void stepper_rot_motor1(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit1, volatile int* rotationCounter1, volatile uint16_t* isStop1)
{
```

```
61      if(dir == STEPPER_CW_MOTOR1)
62      {
63          *isStop1 = 0;
64          *rotationCounter1 += ang;
65      }
66      else if(dir == STEPPER_CCW_MOTOR1)
67      {
68          *isStop1 = 0;
69          *rotationCounter1 -= ang;
70      }
71      stepper_steps_motor1(((((STEPS_PER_REV_MOTOR1 * MICROSTEP_NUM_MOTOR1)/40)*ang)/90),
    stepLimit1);
72      stepper_speed_motor1(rpm);
73      stepper_run_motor1(dir);
74  }
75
76  /* obraca wal silnika do pozycji domowej o zadany kat z zadana predkoscia w kierunku dir */
77  void stepper_rot_home_motor1(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit1,
    volatile int* rotationCounter1, volatile uint16_t* isStop1)
78  {
79      if(*rotationCounter1 < 0)
80      {
81          *isStop1 = 0;
82          uint16_t ang = *rotationCounter1 * (-1);
83          stepper_rot_motor1(ang, rpm, STEPPER_CW_MOTOR1, stepLimit1, rotationCounter1,
    isStop1);
84      }
85      else
86      {
87          *isStop1 = 0;
88          uint16_t ang = *rotationCounter1;
89          stepper_rot_motor1(ang, rpm, STEPPER_CCW_MOTOR1, stepLimit1, rotationCounter1,
    isStop1);
90      }
91  }
92
93
94
```

stepmotor1.h

```
1 #ifndef INC_STEPMOTOR1_H_
2 #define INC_STEPMOTOR1_H_
3
4 #include "stm32f4xx_hal.h"
5
6 #define STEPS_PER_REV_MOTOR1    200             /* liczba krokow na pelen obrot silnika */
7 #define MICROSTEP_NUM_MOTOR1    32              /* liczba mikrokrokow -> 1 = brak
  mikrokrokow */
8
9 #define PULSE_TIM_MOTOR1        htim8           /* timer odpowiedzialny za generowanie
  sygnalu PWM */
10 #define PULSE_TIM_CH_MOTOR1     TIM_CHANNEL_2   /* kanal generowania sygnalu PWM */
11
12 #define XT_TIM_CLK_MOTOR1       84000000        /* czestotliwosc wejsciowa timera PULSE_TIM
  w Hz */
13 #define XT_TIM_PSC_MOTOR1       41              /* prescaler timera PULSE_TIM */
14
15 #define STEPPER_CW_MOTOR1       0               /* stan pinu DIR podczas ruchu CW */
16 #define STEPPER_CCW_MOTOR1      1               /* stan pinu DIR podczas ruchu CCW */
17
18 void stepper_init_motor1(void);
19 void stepper_speed_motor1(uint16_t rpm);
20 void stepper_run_motor1(uint8_t dir);
21 void stepper_stop_motor1(void);
22 void stepper_steps_motor1(uint16_t steps, volatile uint16_t *stepLimit1);
23 void stepper_rot_motor1(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t
  *stepLimit1, volatile int* rotationCounter1, volatile uint16_t* isStop1);
24 void stepper_rot_home_motor1(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit1,
  volatile int* rotationCounter1, volatile uint16_t* isStop1);
25 #endif /* INC_STEPMOTOR1_H_ */
26
```

```c
#include "stepmotor2.h"
#include "tim.h"
#include "gpio.h"
#include "main.h"


/* inicjalizacja biblioteki */
void stepper_init_motor2(void)
{
    HAL_TIM_Base_Start_IT(&PULSE_TIM_MOTOR2);
    HAL_Delay(2);
}

/* ustawia predkosc obrotowa silnika krokowego w rpm */
void stepper_speed_motor2(uint16_t rpm)
{
    if(rpm > 0)
    {
        uint16_t arr_val = XT_TIM_CLK_MOTOR2 / (((XT_TIM_PSC_MOTOR2+1) * rpm * STEPS_PER_REV_MOTOR2 * MICROSTEP_NUM_MOTOR2) / 60) - 1;
        uint16_t pulse_val = arr_val / 2;
        __HAL_TIM_SET_AUTORELOAD(&PULSE_TIM_MOTOR2, arr_val);
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR2, PULSE_TIM_CH_MOTOR2, pulse_val);
    }
    else
    {
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR2, PULSE_TIM_CH_MOTOR2, 0);
    }
}

/* rozpoczyna ruch silnika z kierunkiem obrotu dir */
void stepper_run_motor2(uint8_t dir)
{
    if(dir == STEPPER_CW_MOTOR2)
    {
        HAL_GPIO_WritePin(DIR_AXIS2_GPIO_Port, DIR_AXIS2_Pin, GPIO_PIN_SET);
    }
    else if (dir == STEPPER_CCW_MOTOR2)
    {
        HAL_GPIO_WritePin(DIR_AXIS2_GPIO_Port, DIR_AXIS2_Pin, GPIO_PIN_RESET);
    }
    HAL_Delay(2);
    HAL_TIM_PWM_Start(&PULSE_TIM_MOTOR2, PULSE_TIM_CH_MOTOR2);
}

/* zatrzymuje silnik - wylacza wyjscia kontrolera silnika */
void stepper_stop_motor2(void)
{
    HAL_TIM_PWM_Stop(&PULSE_TIM_MOTOR2, PULSE_TIM_CH_MOTOR2);
}

/* ustawia zadana liczbe krokow do wykonania */
void stepper_steps_motor2(uint16_t steps, volatile uint16_t *stepLimit2)
{
    *stepLimit2 = steps;
}

/* obraca wal silnika o zadny kat z zadana predkoscia w kierunku dir */
void stepper_rot_motor2(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit2, volatile int* rotationCounter2, volatile uint16_t* isStop2)
{
```

```
61      if(dir == STEPPER_CW_MOTOR2)
62      {
63          *isStop2 = 0;
64          *rotationCounter2 += ang;
65      }
66      else if(dir == STEPPER_CCW_MOTOR2)
67      {
68          *isStop2 = 0;
69          *rotationCounter2 -= ang;
70      }
71      stepper_steps_motor2(((((STEPS_PER_REV_MOTOR2 * MICROSTEP_NUM_MOTOR2)/40)*ang)/90),
    stepLimit2);
72      stepper_speed_motor2(rpm);
73      stepper_run_motor2(dir);
74  }
75
76  /* obraca wal silnika do pozycji domowej o zadany kat z zadana predkoscia w kierunku dir */
77  void stepper_rot_home_motor2(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit2,
    volatile int* rotationCounter2, volatile uint16_t* isStop2)
78  {
79      if(*rotationCounter2 < 0)
80      {
81          *isStop2 = 0;
82          uint16_t ang = *rotationCounter2 * (-1);
83          stepper_rot_motor2(ang, rpm, STEPPER_CW_MOTOR2, stepLimit2, rotationCounter2,
    isStop2);
84      }
85      else
86      {
87          *isStop2 = 0;
88          uint16_t ang = *rotationCounter2;
89          stepper_rot_motor2(ang, rpm, STEPPER_CCW_MOTOR2, stepLimit2, rotationCounter2,
    isStop2);
90      }
91  }
92
93
94
```

stepmotor2.h

```
1 #ifndef INC_STEPMOTOR2_H_
2 #define INC_STEPMOTOR2_H_
3
4 #include "stm32f4xx_hal.h"
5
6 #define STEPS_PER_REV_MOTOR2    200             /* liczba krokow na pelen obrot silnika */
7 #define MICROSTEP_NUM_MOTOR2    32              /* liczba mikrokrokow -> 1 = brak
  mikrokrokow */
8
9 #define PULSE_TIM_MOTOR2        htim4           /* timer odpowiedzialny za generowanie
  sygnalu PWM */
10 #define PULSE_TIM_CH_MOTOR2     TIM_CHANNEL_1   /* kanal generowania sygnalu PWM */
11
12 #define XT_TIM_CLK_MOTOR2       84000000        /* czestotliwosc wejsciowa timera PULSE_TIM
  w Hz */
13 #define XT_TIM_PSC_MOTOR2       41              /* prescaler timera PULSE_TIM */
14
15 #define STEPPER_CW_MOTOR2       0               /* stan pinu DIR podczas ruchu CW */
16 #define STEPPER_CCW_MOTOR2      1               /* stan pinu DIR podczas ruchu CCW */
17
18 void stepper_init_motor2(void);
19 void stepper_speed_motor2(uint16_t rpm);
20 void stepper_run_motor2(uint8_t dir);
21 void stepper_stop_motor2(void);
22 void stepper_steps_motor2(uint16_t steps, volatile uint16_t *stepLimit2);
23 void stepper_rot_motor2(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t
  *stepLimit2, volatile int* rotationCounter2, volatile uint16_t* isStop2);
24 void stepper_rot_home_motor2(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit2,
  volatile int* rotationCounter2, volatile uint16_t* isStop2);
25 #endif /* INC_STEPMOTOR2_H_ */
26
```

```c
#include "stepmotor3.h"
#include "tim.h"
#include "gpio.h"
#include "main.h"


/* inicjalizacja biblioteki */
void stepper_init_motor3(void)
{
    HAL_TIM_Base_Start_IT(&PULSE_TIM_MOTOR3);
    HAL_Delay(2);
}

/* ustawia predkosc obrotowa silnika krokowego w rpm */
void stepper_speed_motor3(uint16_t rpm)
{
    if(rpm > 0)
    {
        uint16_t arr_val = XT_TIM_CLK_MOTOR3 / (((XT_TIM_PSC_MOTOR3+1) * rpm * STEPS_PER_REV_MOTOR3 * MICROSTEP_NUM_MOTOR3) / 60) - 1;
        uint16_t pulse_val = arr_val / 2;
        __HAL_TIM_SET_AUTORELOAD(&PULSE_TIM_MOTOR3, arr_val);
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR3, PULSE_TIM_CH_MOTOR3, pulse_val);
    }
    else
    {
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR3, PULSE_TIM_CH_MOTOR3, 0);
    }
}

/* rozpoczyna ruch silnika z kierunkiem obrotu dir */
void stepper_run_motor3(uint8_t dir)
{
    if(dir == STEPPER_CW_MOTOR3)
    {
        HAL_GPIO_WritePin(DIR_AXIS3_GPIO_Port, DIR_AXIS3_Pin, GPIO_PIN_SET);
    }
    else if (dir == STEPPER_CCW_MOTOR3)
    {
        HAL_GPIO_WritePin(DIR_AXIS3_GPIO_Port, DIR_AXIS3_Pin, GPIO_PIN_RESET);
    }
    HAL_Delay(2);
    HAL_TIM_PWM_Start(&PULSE_TIM_MOTOR3, PULSE_TIM_CH_MOTOR3);
}

/* zatrzymuje silnik - wylacza wyjscia kontrolera silnika */
void stepper_stop_motor3(void)
{
    HAL_TIM_PWM_Stop(&PULSE_TIM_MOTOR3, PULSE_TIM_CH_MOTOR3);
}

/* ustawia zadana liczbe krokow do wykonania */
void stepper_steps_motor3(uint16_t steps, volatile uint16_t *stepLimit3)
{
    *stepLimit3 = steps;
}

/* obraca wal silnika o zadny kat z zadana predkoscia w kierunku dir */
void stepper_rot_motor3(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit3, volatile int* rotationCounter3, volatile uint16_t* isStop3)
{
```

```
61      if(dir == STEPPER_CW_MOTOR3)
62      {
63          *isStop3 = 0;
64          *rotationCounter3 += ang;
65      }
66      else if(dir == STEPPER_CCW_MOTOR3)
67      {
68          *isStop3 = 0;
69          *rotationCounter3 -= ang;
70      }
71      stepper_steps_motor3(((((STEPS_PER_REV_MOTOR3 * MICROSTEP_NUM_MOTOR3)/40)*ang)/90),
    stepLimit3);
72      stepper_speed_motor3(rpm);
73      stepper_run_motor3(dir);
74  }
75
76  /* obraca wal silnika do pozycji domowej o zadany kat z zadana predkoscia w kierunku dir */
77  void stepper_rot_home_motor3(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit3,
    volatile int* rotationCounter3, volatile uint16_t* isStop3)
78  {
79      if(*rotationCounter3 < 0)
80      {
81          *isStop3 = 0;
82          uint16_t ang = *rotationCounter3 * (-1);
83          stepper_rot_motor3(ang, rpm, STEPPER_CW_MOTOR3, stepLimit3, rotationCounter3,
    isStop3);
84      }
85      else
86      {
87          *isStop3 = 0;
88          uint16_t ang = *rotationCounter3;
89          stepper_rot_motor3(ang, rpm, STEPPER_CCW_MOTOR3, stepLimit3, rotationCounter3,
    isStop3);
90      }
91  }
92
93
94
```

stepmotor3.h

```
#ifndef INC_STEPMOTOR3_H_
#define INC_STEPMOTOR3_H_

#include "stm32f4xx_hal.h"

#define STEPS_PER_REV_MOTOR3    200             /* liczba krokow na pelen obrot silnika */
#define MICROSTEP_NUM_MOTOR3    32              /* liczba mikrokrokow -> 1 = brak
 mikrokrokow */

#define PULSE_TIM_MOTOR3        htim14          /* timer odpowiedzialny za generowanie
 sygnalu PWM */
#define PULSE_TIM_CH_MOTOR3     TIM_CHANNEL_1   /* kanal generowania sygnalu PWM */

#define XT_TIM_CLK_MOTOR3       84000000        /* czestotliwosc wejsciowa timera PULSE_TIM
 w Hz */
#define XT_TIM_PSC_MOTOR3       41              /* prescaler timera PULSE_TIM */

#define STEPPER_CW_MOTOR3       0               /* stan pinu DIR podczas ruchu CW */
#define STEPPER_CCW_MOTOR3      1               /* stan pinu DIR podczas ruchu CCW */

void stepper_init_motor3(void);
void stepper_speed_motor3(uint16_t rpm);
void stepper_run_motor3(uint8_t dir);
void stepper_stop_motor3(void);
void stepper_steps_motor3(uint16_t steps, volatile uint16_t *stepLimit3);
void stepper_rot_motor3(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t
 *stepLimit3, volatile int* rotationCounter3, volatile uint16_t* isStop3);
void stepper_rot_home_motor3(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit3,
 volatile int* rotationCounter3, volatile uint16_t* isStop3);
#endif /* INC_STEPMOTOR3_H_ */

```

```c
#include "stepmotor4.h"
#include "tim.h"
#include "gpio.h"
#include "main.h"


/* inicjalizacja biblioteki */
void stepper_init_motor4(void)
{
    HAL_TIM_Base_Start_IT(&PULSE_TIM_MOTOR4);
    HAL_Delay(2);
}

/* ustawia predkosc obrotowa silnika krokowego w rpm */
void stepper_speed_motor4(uint16_t rpm)
{
    if(rpm > 0)
    {
        uint16_t arr_val = XT_TIM_CLK_MOTOR4 / (((XT_TIM_PSC_MOTOR4+1) * rpm * STEPS_PER_REV_MOTOR4 * MICROSTEP_NUM_MOTOR4) / 60) - 1;
        uint16_t pulse_val = arr_val / 2;
        __HAL_TIM_SET_AUTORELOAD(&PULSE_TIM_MOTOR4, arr_val);
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR4, PULSE_TIM_CH_MOTOR4, pulse_val);
    }
    else
    {
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR4, PULSE_TIM_CH_MOTOR4, 0);
    }
}

/* rozpoczyna ruch silnika z kierunkiem obrotu dir */
void stepper_run_motor4(uint8_t dir)
{
    if(dir == STEPPER_CW_MOTOR4)
    {
        HAL_GPIO_WritePin(DIR_AXIS4_GPIO_Port, DIR_AXIS4_Pin, GPIO_PIN_SET);
    }
    else if (dir == STEPPER_CCW_MOTOR4)
    {
        HAL_GPIO_WritePin(DIR_AXIS4_GPIO_Port, DIR_AXIS4_Pin, GPIO_PIN_RESET);
    }
    HAL_Delay(2);
    HAL_TIM_PWM_Start(&PULSE_TIM_MOTOR4, PULSE_TIM_CH_MOTOR4);
}

/* zatrzymuje silnik - wylacza wyjscia kontrolera silnika */
void stepper_stop_motor4(void)
{
    HAL_TIM_PWM_Stop(&PULSE_TIM_MOTOR4, PULSE_TIM_CH_MOTOR4);
}

/* ustawia zadana liczbe krokow do wykonania */
void stepper_steps_motor4(uint16_t steps, volatile uint16_t *stepLimit4)
{
    *stepLimit4 = steps;
}

/* obraca wal silnika o zadny kat z zadana predkoscia w kierunku dir */
void stepper_rot_motor4(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit4, volatile int* rotationCounter4, volatile uint16_t* isStop4)
{
```

```
61      if(dir == STEPPER_CW_MOTOR4)
62      {
63          *isStop4 = 0;
64          *rotationCounter4 += ang;
65      }
66      else if(dir == STEPPER_CCW_MOTOR4)
67      {
68          *isStop4 = 0;
69          *rotationCounter4 -= ang;
70      }
71      stepper_steps_motor4(((((STEPS_PER_REV_MOTOR4 * MICROSTEP_NUM_MOTOR4)/40)*ang)/90),
    stepLimit4);
72      stepper_speed_motor4(rpm);
73      stepper_run_motor4(dir);
74  }
75
76  /* obraca wal silnika do pozycji domowej o zadany kat z zadana predkoscia w kierunku dir */
77  void stepper_rot_home_motor4(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit4,
    volatile int* rotationCounter4, volatile uint16_t* isStop4)
78  {
79      if(*rotationCounter4 < 0)
80      {
81          *isStop4 = 0;
82          uint16_t ang = *rotationCounter4 * (-1);
83          stepper_rot_motor4(ang, rpm, STEPPER_CW_MOTOR4, stepLimit4, rotationCounter4,
    isStop4);
84      }
85      else
86      {
87          *isStop4 = 0;
88          uint16_t ang = *rotationCounter4;
89          stepper_rot_motor4(ang, rpm, STEPPER_CCW_MOTOR4, stepLimit4, rotationCounter4,
    isStop4);
90      }
91  }
92
93
94
```

stepmotor4.h

```
#ifndef INC_STEPMOTOR4_H_
#define INC_STEPMOTOR4_H_

#include "stm32f4xx_hal.h"

#define STEPS_PER_REV_MOTOR4    200             /* liczba krokow na pelen obrot silnika */
#define MICROSTEP_NUM_MOTOR4    32              /* liczba mikrokrokow -> 1 = brak
mikrokrokow */

#define PULSE_TIM_MOTOR4        htim1           /* timer odpowiedzialny za generowanie
sygnalu PWM */
#define PULSE_TIM_CH_MOTOR4     TIM_CHANNEL_3   /* kanal generowania sygnalu PWM */

#define XT_TIM_CLK_MOTOR4       84000000        /* czestotliwosc wejsciowa timera PULSE_TIM
w Hz */
#define XT_TIM_PSC_MOTOR4       41              /* prescaler timera PULSE_TIM */

#define STEPPER_CW_MOTOR4       0               /* stan pinu DIR podczas ruchu CW */
#define STEPPER_CCW_MOTOR4      1               /* stan pinu DIR podczas ruchu CCW */

void stepper_init_motor4(void);
void stepper_speed_motor4(uint16_t rpm);
void stepper_run_motor4(uint8_t dir);
void stepper_stop_motor4(void);
void stepper_steps_motor4(uint16_t steps, volatile uint16_t *stepLimit4);
void stepper_rot_motor4(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t
*stepLimit4, volatile int* rotationCounter4, volatile uint16_t* isStop4);
void stepper_rot_home_motor4(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit4,
volatile int* rotationCounter4, volatile uint16_t* isStop4);
#endif /* INC_STEPMOTOR4_H_ */

```

```c
#include "stepmotor5.h"
#include "tim.h"
#include "gpio.h"
#include "main.h"


/* inicjalizacja biblioteki */
void stepper_init_motor5(void)
{
    HAL_TIM_Base_Start_IT(&PULSE_TIM_MOTOR5);
    HAL_Delay(2);
}

/* ustawia predkosc obrotowa silnika krokowego w rpm */
void stepper_speed_motor5(uint16_t rpm)
{
    if(rpm > 0)
    {
        uint16_t arr_val = XT_TIM_CLK_MOTOR5 / (((XT_TIM_PSC_MOTOR5+1) * rpm * STEPS_PER_REV_MOTOR5 * MICROSTEP_NUM_MOTOR5) / 60) - 1;
        uint16_t pulse_val = arr_val / 2;
        __HAL_TIM_SET_AUTORELOAD(&PULSE_TIM_MOTOR5, arr_val);
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR5, PULSE_TIM_CH_MOTOR5, pulse_val);
    }
    else
    {
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR5, PULSE_TIM_CH_MOTOR5, 0);
    }
}

/* rozpoczyna ruch silnika z kierunkiem obrotu dir */
void stepper_run_motor5(uint8_t dir)
{
    if(dir == STEPPER_CW_MOTOR5)
    {
        HAL_GPIO_WritePin(DIR_AXIS5_GPIO_Port, DIR_AXIS5_Pin, GPIO_PIN_SET);
    }
    else if (dir == STEPPER_CCW_MOTOR5)
    {
        HAL_GPIO_WritePin(DIR_AXIS5_GPIO_Port, DIR_AXIS5_Pin, GPIO_PIN_RESET);
    }
    HAL_Delay(2);
    HAL_TIM_PWM_Start(&PULSE_TIM_MOTOR5, PULSE_TIM_CH_MOTOR5);
}

/* zatrzymuje silnik - wylacza wyjscia kontrolera silnika */
void stepper_stop_motor5(void)
{
    HAL_TIM_PWM_Stop(&PULSE_TIM_MOTOR5, PULSE_TIM_CH_MOTOR5);
}

/* ustawia zadana liczbe krokow do wykonania */
void stepper_steps_motor5(uint16_t steps, volatile uint16_t *stepLimit5)
{
    *stepLimit5 = steps;
}

/* obraca wal silnika o zadny kat z zadana predkoscia w kierunku dir */
void stepper_rot_motor5(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit5, volatile int* rotationCounter5, volatile uint16_t* isStop5)
{
```

```
61      if(dir == STEPPER_CW_MOTOR5)
62      {
63          *isStop5 = 0;
64          *rotationCounter5 += ang;
65      }
66      else if(dir == STEPPER_CCW_MOTOR5)
67      {
68          *isStop5 = 0;
69          *rotationCounter5 -= ang;
70      }
71      stepper_steps_motor5(((((STEPS_PER_REV_MOTOR5 * MICROSTEP_NUM_MOTOR5)/40)*ang)/90),
    stepLimit5);
72      stepper_speed_motor5(rpm);
73      stepper_run_motor5(dir);
74  }
75
76  /* obraca wal silnika do pozycji domowej o zadany kat z zadana predkoscia w kierunku dir */
77  void stepper_rot_home_motor5(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit5,
    volatile int* rotationCounter5, volatile uint16_t* isStop5)
78  {
79      if(*rotationCounter5 < 0)
80      {
81          *isStop5 = 0;
82          uint16_t ang = *rotationCounter5 * (-1);
83          stepper_rot_motor5(ang, rpm, STEPPER_CW_MOTOR5, stepLimit5, rotationCounter5,
    isStop5);
84      }
85      else
86      {
87          *isStop5 = 0;
88          uint16_t ang = *rotationCounter5;
89          stepper_rot_motor5(ang, rpm, STEPPER_CCW_MOTOR5, stepLimit5, rotationCounter5,
    isStop5);
90      }
91  }
92
93
94
```

stepmotor5.h

```
1 #ifndef INC_STEPMOTOR5_H_
2 #define INC_STEPMOTOR5_H_
3
4 #include "stm32f4xx_hal.h"
5
6 #define STEPS_PER_REV_MOTOR5    200             /* liczba krokow na pelen obrot silnika */
7 #define MICROSTEP_NUM_MOTOR5    32              /* liczba mikrokrokow -> 1 = brak
  mikrokrokow */
8
9 #define PULSE_TIM_MOTOR5        htim2           /* timer odpowiedzialny za generowanie
  sygnalu PWM */
10 #define PULSE_TIM_CH_MOTOR5     TIM_CHANNEL_2   /* kanal generowania sygnalu PWM */
11
12 #define XT_TIM_CLK_MOTOR5       84000000        /* czestotliwosc wejsciowa timera PULSE_TIM
  w Hz */
13 #define XT_TIM_PSC_MOTOR5       41              /* prescaler timera PULSE_TIM */
14
15 #define STEPPER_CW_MOTOR5       0               /* stan pinu DIR podczas ruchu CW */
16 #define STEPPER_CCW_MOTOR5      1               /* stan pinu DIR podczas ruchu CCW */
17
18 void stepper_init_motor5(void);
19 void stepper_speed_motor5(uint16_t rpm);
20 void stepper_run_motor5(uint8_t dir);
21 void stepper_stop_motor5(void);
22 void stepper_steps_motor5(uint16_t steps, volatile uint16_t *stepLimit5);
23 void stepper_rot_motor5(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t
  *stepLimit5, volatile int* rotationCounter5, volatile uint16_t* isStop5);
24 void stepper_rot_home_motor5(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit5,
  volatile int* rotationCounter5, volatile uint16_t* isStop5);
25 #endif /* INC_STEPMOTOR5_H_ */
26
```

```c
#include "stepmotor6.h"
#include "tim.h"
#include "gpio.h"
#include "main.h"


/* inicjalizacja biblioteki */
void stepper_init_motor6(void)
{
    HAL_TIM_Base_Start_IT(&PULSE_TIM_MOTOR6);
    HAL_Delay(2);
}

/* ustawia predkosc obrotowa silnika krokowego w rpm */
void stepper_speed_motor6(uint16_t rpm)
{
    if(rpm > 0)
    {
        uint16_t arr_val = XT_TIM_CLK_MOTOR6 / (((XT_TIM_PSC_MOTOR6+1) * rpm * STEPS_PER_REV_MOTOR6 * MICROSTEP_NUM_MOTOR6) / 60) - 1;
        uint16_t pulse_val = arr_val / 2;
        __HAL_TIM_SET_AUTORELOAD(&PULSE_TIM_MOTOR6, arr_val);
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR6, PULSE_TIM_CH_MOTOR6, pulse_val);
    }
    else
    {
        __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR6, PULSE_TIM_CH_MOTOR6, 0);
    }
}

/* rozpoczyna ruch silnika z kierunkiem obrotu dir */
void stepper_run_motor6(uint8_t dir)
{
    if(dir == STEPPER_CW_MOTOR6)
    {
        HAL_GPIO_WritePin(DIR_AXIS6_GPIO_Port, DIR_AXIS6_Pin, GPIO_PIN_SET);
    }
    else if (dir == STEPPER_CCW_MOTOR6)
    {
        HAL_GPIO_WritePin(DIR_AXIS6_GPIO_Port, DIR_AXIS6_Pin, GPIO_PIN_RESET);
    }
    HAL_Delay(2);
    HAL_TIM_PWM_Start(&PULSE_TIM_MOTOR6, PULSE_TIM_CH_MOTOR6);
}

/* zatrzymuje silnik - wylacza wyjscia kontrolera silnika */
void stepper_stop_motor6(void)
{
    HAL_TIM_PWM_Stop(&PULSE_TIM_MOTOR6, PULSE_TIM_CH_MOTOR6);
}

/* ustawia zadana liczbe krokow do wykonania */
void stepper_steps_motor6(uint16_t steps, volatile uint16_t *stepLimit6)
{
    *stepLimit6 = steps;
}

/* obraca wal silnika o zadny kat z zadana predkoscia w kierunku dir */
void stepper_rot_motor6(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit6, volatile int* rotationCounter6, volatile uint16_t* isStop6)
{
```

```
61      if(dir == STEPPER_CW_MOTOR6)
62      {
63          *isStop6 = 0;
64          *rotationCounter6 += ang;
65      }
66      else if(dir == STEPPER_CCW_MOTOR6)
67      {
68          *isStop6 = 0;
69          *rotationCounter6 -= ang;
70      }
71      stepper_steps_motor6(((((STEPS_PER_REV_MOTOR6 * MICROSTEP_NUM_MOTOR6)/40)*ang)/90),
    stepLimit6);
72      stepper_speed_motor6(rpm);
73      stepper_run_motor6(dir);
74  }
75
76  /* obraca wal silnika do pozycji domowej o zadany kat z zadana predkoscia w kierunku dir */
77  void stepper_rot_home_motor6(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit6,
    volatile int* rotationCounter6, volatile uint16_t* isStop6)
78  {
79      if(*rotationCounter6 < 0)
80      {
81          *isStop6 = 0;
82          uint16_t ang = *rotationCounter6 * (-1);
83          stepper_rot_motor6(ang, rpm, STEPPER_CW_MOTOR6, stepLimit6, rotationCounter6,
    isStop6);
84      }
85      else
86      {
87          *isStop6 = 0;
88          uint16_t ang = *rotationCounter6;
89          stepper_rot_motor6(ang, rpm, STEPPER_CCW_MOTOR6, stepLimit6, rotationCounter6,
    isStop6);
90      }
91  }
92
93
94
```

stepmotor6.h

```
1 #ifndef INC_STEPMOTOR6_H_
2 #define INC_STEPMOTOR6_H_
3
4 #include "stm32f4xx_hal.h"
5
6 #define STEPS_PER_REV_MOTOR6    200             /* liczba krokow na pelen obrot silnika */
7 #define MICROSTEP_NUM_MOTOR6    32              /* liczba mikrokrokow -> 1 = brak
  mikrokrokow */
8
9 #define PULSE_TIM_MOTOR6        htim3           /* timer odpowiedzialny za generowanie
  sygnalu PWM */
10 #define PULSE_TIM_CH_MOTOR6     TIM_CHANNEL_2   /* kanal generowania sygnalu PWM */
11
12 #define XT_TIM_CLK_MOTOR6       84000000        /* czestotliwosc wejsciowa timera PULSE_TIM
  w Hz */
13 #define XT_TIM_PSC_MOTOR6       41              /* prescaler timera PULSE_TIM */
14
15 #define STEPPER_CW_MOTOR6       0               /* stan pinu DIR podczas ruchu CW */
16 #define STEPPER_CCW_MOTOR6      1               /* stan pinu DIR podczas ruchu CCW */
17
18 void stepper_init_motor6(void);
19 void stepper_speed_motor6(uint16_t rpm);
20 void stepper_run_motor6(uint8_t dir);
21 void stepper_stop_motor6(void);
22 void stepper_steps_motor6(uint16_t steps, volatile uint16_t *stepLimit6);
23 void stepper_rot_motor6(uint16_t ang, uint16_t rpm, uint8_t dir, volatile uint16_t
  *stepLimit6, volatile int* rotationCounter6, volatile uint16_t* isStop6);
24 void stepper_rot_home_motor6(uint16_t rpm, uint8_t dir, volatile uint16_t *stepLimit6,
  volatile int* rotationCounter6, volatile uint16_t* isStop6);
25 #endif /* INC_STEPMOTOR6_H_ */
26
```

```c
#include "servomotor7.h"
#include "tim.h"
#include "gpio.h"
#include "main.h"

/* inicjalizacja biblioteki */
void servo_init_motor7(void)
{
    HAL_TIM_PWM_Start(&PULSE_TIM_MOTOR7, PULSE_TIM_CH_MOTOR7);
    HAL_Delay(2);
    __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR7, PULSE_TIM_CH_MOTOR7, OPEN_GRIPPER);
}

void servo_open_gripper(void)
{
    __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR7, PULSE_TIM_CH_MOTOR7, OPEN_GRIPPER);
}

void servo_close_gripper(void)
{
    __HAL_TIM_SET_COMPARE(&PULSE_TIM_MOTOR7, PULSE_TIM_CH_MOTOR7, CLOSE_GRIPPER);
}

```

servomotor7.h

```
1 #ifndef INC_SERVOMOTOR7_H_
2 #define INC_SERVOMOTOR7_H_
3
4 #include "stm32f4xx_hal.h"
5
6 #define PULSE_TIM_MOTOR7        htim10          /* timer odpowiedzialny za generowanie
  sygnalu PWM */
7 #define PULSE_TIM_CH_MOTOR7     TIM_CHANNEL_1   /* kanal generowania sygnalu PWM */
8
9 #define XT_TIM_CLK_MOTOR7       84000000        /* czestotliwosc wejsciowa timera PULSE_TIM
  w Hz */
10 #define XT_TIM_PSC_MOTOR7       83              /* prescaler timera PULSE_TIM */
11
12 #define OPEN_GRIPPER            900         /* szerokosc impulsu odpowiedzialna za otwarcie
  chwytaka */
13 #define CLOSE_GRIPPER           1500            /* szerokosc impulsu odpowiedzialna za
  otwarcie chwytaka */
14
15 void servo_init_motor7(void);
16 void servo_open_gripper(void);
17 void servo_close_gripper(void);
18 #endif /* INC_SERVOMOTOR7_H_ */
19
```

```c
/**
  ******************************************************************************
  * File Name          : TIM.c
  * Description        : This file provides code for the configuration
  *                      of the TIM instances.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "tim.h"

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;
TIM_HandleTypeDef htim8;
TIM_HandleTypeDef htim10;
TIM_HandleTypeDef htim14;

/* TIM1 init function */
void MX_TIM1_Init(void)
{
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};
  TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};

  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 41;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 9999;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
```

```
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 5000;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_3) != HAL_OK)
  {
    Error_Handler();
  }
  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig) != HAL_OK)
  {
    Error_Handler();
  }
  HAL_TIM_MspPostInit(&htim1);

}
/* TIM2 init function */
void MX_TIM2_Init(void)
{
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 41;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 9999;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
```

```
125   }
126   sConfigOC.OCMode = TIM_OCMODE_PWM1;
127   sConfigOC.Pulse = 5000;
128   sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
129   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
130   if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
131   {
132     Error_Handler();
133   }
134   HAL_TIM_MspPostInit(&htim2);
135
136 }
137 /* TIM3 init function */
138 void MX_TIM3_Init(void)
139 {
140   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
141   TIM_MasterConfigTypeDef sMasterConfig = {0};
142   TIM_OC_InitTypeDef sConfigOC = {0};
143
144   htim3.Instance = TIM3;
145   htim3.Init.Prescaler = 41;
146   htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
147   htim3.Init.Period = 9999;
148   htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
149   htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
150   if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
151   {
152     Error_Handler();
153   }
154   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
155   if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
156   {
157     Error_Handler();
158   }
159   if (HAL_TIM_PWM_Init(&htim3) != HAL_OK)
160   {
161     Error_Handler();
162   }
163   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
164   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
165   if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
166   {
167     Error_Handler();
168   }
169   sConfigOC.OCMode = TIM_OCMODE_PWM1;
170   sConfigOC.Pulse = 5000;
171   sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
172   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
173   if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
174   {
175     Error_Handler();
176   }
177   HAL_TIM_MspPostInit(&htim3);
178
179 }
180 /* TIM4 init function */
181 void MX_TIM4_Init(void)
182 {
183   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
184   TIM_MasterConfigTypeDef sMasterConfig = {0};
185   TIM_OC_InitTypeDef sConfigOC = {0};
186
```

```
187   htim4.Instance = TIM4;
188   htim4.Init.Prescaler = 41;
189   htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
190   htim4.Init.Period = 9999;
191   htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
192   htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
193   if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
194   {
195     Error_Handler();
196   }
197   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
198   if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
199   {
200     Error_Handler();
201   }
202   if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
203   {
204     Error_Handler();
205   }
206   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
207   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
208   if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
209   {
210     Error_Handler();
211   }
212   sConfigOC.OCMode = TIM_OCMODE_PWM1;
213   sConfigOC.Pulse = 5000;
214   sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
215   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
216   if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
217   {
218     Error_Handler();
219   }
220   HAL_TIM_MspPostInit(&htim4);
221
222 }
223 /* TIM8 init function */
224 void MX_TIM8_Init(void)
225 {
226   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
227   TIM_MasterConfigTypeDef sMasterConfig = {0};
228   TIM_OC_InitTypeDef sConfigOC = {0};
229   TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};
230
231   htim8.Instance = TIM8;
232   htim8.Init.Prescaler = 41;
233   htim8.Init.CounterMode = TIM_COUNTERMODE_UP;
234   htim8.Init.Period = 9999;
235   htim8.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
236   htim8.Init.RepetitionCounter = 0;
237   htim8.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
238   if (HAL_TIM_Base_Init(&htim8) != HAL_OK)
239   {
240     Error_Handler();
241   }
242   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
243   if (HAL_TIM_ConfigClockSource(&htim8, &sClockSourceConfig) != HAL_OK)
244   {
245     Error_Handler();
246   }
247   if (HAL_TIM_PWM_Init(&htim8) != HAL_OK)
248   {
```

```
249     Error_Handler();
250   }
251   sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
252   sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
253   if (HAL_TIMEx_MasterConfigSynchronization(&htim8, &sMasterConfig) != HAL_OK)
254   {
255     Error_Handler();
256   }
257   sConfigOC.OCMode = TIM_OCMODE_PWM1;
258   sConfigOC.Pulse = 5000;
259   sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
260   sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
261   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
262   sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
263   sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
264   if (HAL_TIM_PWM_ConfigChannel(&htim8, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
265   {
266     Error_Handler();
267   }
268   sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
269   sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
270   sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
271   sBreakDeadTimeConfig.DeadTime = 0;
272   sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
273   sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
274   sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
275   if (HAL_TIMEx_ConfigBreakDeadTime(&htim8, &sBreakDeadTimeConfig) != HAL_OK)
276   {
277     Error_Handler();
278   }
279   HAL_TIM_MspPostInit(&htim8);
280
281 }
282 /* TIM10 init function */
283 void MX_TIM10_Init(void)
284 {
285   TIM_OC_InitTypeDef sConfigOC = {0};
286
287   htim10.Instance = TIM10;
288   htim10.Init.Prescaler = 81;
289   htim10.Init.CounterMode = TIM_COUNTERMODE_UP;
290   htim10.Init.Period = 19999;
291   htim10.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
292   htim10.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
293   if (HAL_TIM_Base_Init(&htim10) != HAL_OK)
294   {
295     Error_Handler();
296   }
297   if (HAL_TIM_PWM_Init(&htim10) != HAL_OK)
298   {
299     Error_Handler();
300   }
301   sConfigOC.OCMode = TIM_OCMODE_PWM1;
302   sConfigOC.Pulse = 0;
303   sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
304   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
305   if (HAL_TIM_PWM_ConfigChannel(&htim10, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
306   {
307     Error_Handler();
308   }
309   HAL_TIM_MspPostInit(&htim10);
310
```

```
311 }
312 /* TIM14 init function */
313 void MX_TIM14_Init(void)
314 {
315   TIM_OC_InitTypeDef sConfigOC = {0};
316
317   htim14.Instance = TIM14;
318   htim14.Init.Prescaler = 41;
319   htim14.Init.CounterMode = TIM_COUNTERMODE_UP;
320   htim14.Init.Period = 9999;
321   htim14.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
322   htim14.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
323   if (HAL_TIM_Base_Init(&htim14) != HAL_OK)
324   {
325     Error_Handler();
326   }
327   if (HAL_TIM_PWM_Init(&htim14) != HAL_OK)
328   {
329     Error_Handler();
330   }
331   sConfigOC.OCMode = TIM_OCMODE_PWM1;
332   sConfigOC.Pulse = 5000;
333   sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
334   sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
335   if (HAL_TIM_PWM_ConfigChannel(&htim14, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
336   {
337     Error_Handler();
338   }
339   HAL_TIM_MspPostInit(&htim14);
340
341 }
342
343 void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* tim_baseHandle)
344 {
345
346   if(tim_baseHandle->Instance==TIM1)
347   {
348   /* USER CODE BEGIN TIM1_MspInit 0 */
349
350   /* USER CODE END TIM1_MspInit 0 */
351     /* TIM1 clock enable */
352     __HAL_RCC_TIM1_CLK_ENABLE();
353
354     /* TIM1 interrupt Init */
355     HAL_NVIC_SetPriority(TIM1_UP_TIM10_IRQn, 0, 0);
356     HAL_NVIC_EnableIRQ(TIM1_UP_TIM10_IRQn);
357   /* USER CODE BEGIN TIM1_MspInit 1 */
358
359   /* USER CODE END TIM1_MspInit 1 */
360   }
361   else if(tim_baseHandle->Instance==TIM2)
362   {
363   /* USER CODE BEGIN TIM2_MspInit 0 */
364
365   /* USER CODE END TIM2_MspInit 0 */
366     /* TIM2 clock enable */
367     __HAL_RCC_TIM2_CLK_ENABLE();
368
369     /* TIM2 interrupt Init */
370     HAL_NVIC_SetPriority(TIM2_IRQn, 0, 0);
371     HAL_NVIC_EnableIRQ(TIM2_IRQn);
372   /* USER CODE BEGIN TIM2_MspInit 1 */
```

```
373
374   /* USER CODE END TIM2_MspInit 1 */
375   }
376   else if(tim_baseHandle->Instance==TIM3)
377   {
378   /* USER CODE BEGIN TIM3_MspInit 0 */
379
380   /* USER CODE END TIM3_MspInit 0 */
381     /* TIM3 clock enable */
382     __HAL_RCC_TIM3_CLK_ENABLE();
383
384     /* TIM3 interrupt Init */
385     HAL_NVIC_SetPriority(TIM3_IRQn, 0, 0);
386     HAL_NVIC_EnableIRQ(TIM3_IRQn);
387   /* USER CODE BEGIN TIM3_MspInit 1 */
388
389   /* USER CODE END TIM3_MspInit 1 */
390   }
391   else if(tim_baseHandle->Instance==TIM4)
392   {
393   /* USER CODE BEGIN TIM4_MspInit 0 */
394
395   /* USER CODE END TIM4_MspInit 0 */
396     /* TIM4 clock enable */
397     __HAL_RCC_TIM4_CLK_ENABLE();
398
399     /* TIM4 interrupt Init */
400     HAL_NVIC_SetPriority(TIM4_IRQn, 0, 0);
401     HAL_NVIC_EnableIRQ(TIM4_IRQn);
402   /* USER CODE BEGIN TIM4_MspInit 1 */
403
404   /* USER CODE END TIM4_MspInit 1 */
405   }
406   else if(tim_baseHandle->Instance==TIM8)
407   {
408   /* USER CODE BEGIN TIM8_MspInit 0 */
409
410   /* USER CODE END TIM8_MspInit 0 */
411     /* TIM8 clock enable */
412     __HAL_RCC_TIM8_CLK_ENABLE();
413
414     /* TIM8 interrupt Init */
415     HAL_NVIC_SetPriority(TIM8_UP_TIM13_IRQn, 0, 0);
416     HAL_NVIC_EnableIRQ(TIM8_UP_TIM13_IRQn);
417     HAL_NVIC_SetPriority(TIM8_TRG_COM_TIM14_IRQn, 0, 0);
418     HAL_NVIC_EnableIRQ(TIM8_TRG_COM_TIM14_IRQn);
419   /* USER CODE BEGIN TIM8_MspInit 1 */
420
421   /* USER CODE END TIM8_MspInit 1 */
422   }
423   else if(tim_baseHandle->Instance==TIM10)
424   {
425   /* USER CODE BEGIN TIM10_MspInit 0 */
426
427   /* USER CODE END TIM10_MspInit 0 */
428     /* TIM10 clock enable */
429     __HAL_RCC_TIM10_CLK_ENABLE();
430
431     /* TIM10 interrupt Init */
432     HAL_NVIC_SetPriority(TIM1_UP_TIM10_IRQn, 0, 0);
433     HAL_NVIC_EnableIRQ(TIM1_UP_TIM10_IRQn);
434   /* USER CODE BEGIN TIM10_MspInit 1 */
```

```
435
436   /* USER CODE END TIM10_MspInit 1 */
437   }
438   else if(tim_baseHandle->Instance==TIM14)
439   {
440   /* USER CODE BEGIN TIM14_MspInit 0 */
441
442   /* USER CODE END TIM14_MspInit 0 */
443     /* TIM14 clock enable */
444     __HAL_RCC_TIM14_CLK_ENABLE();
445
446     /* TIM14 interrupt Init */
447     HAL_NVIC_SetPriority(TIM8_TRG_COM_TIM14_IRQn, 0, 0);
448     HAL_NVIC_EnableIRQ(TIM8_TRG_COM_TIM14_IRQn);
449   /* USER CODE BEGIN TIM14_MspInit 1 */
450
451   /* USER CODE END TIM14_MspInit 1 */
452   }
453 }
454 void HAL_TIM_MspPostInit(TIM_HandleTypeDef* timHandle)
455 {
456
457   GPIO_InitTypeDef GPIO_InitStruct = {0};
458   if(timHandle->Instance==TIM1)
459   {
460   /* USER CODE BEGIN TIM1_MspPostInit 0 */
461
462   /* USER CODE END TIM1_MspPostInit 0 */
463     __HAL_RCC_GPIOA_CLK_ENABLE();
464     /**TIM1 GPIO Configuration
465     PA10     ------> TIM1_CH3
466     */
467     GPIO_InitStruct.Pin = GPIO_PIN_10;
468     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
469     GPIO_InitStruct.Pull = GPIO_NOPULL;
470     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
471     GPIO_InitStruct.Alternate = GPIO_AF1_TIM1;
472     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
473
474   /* USER CODE BEGIN TIM1_MspPostInit 1 */
475
476   /* USER CODE END TIM1_MspPostInit 1 */
477   }
478   else if(timHandle->Instance==TIM2)
479   {
480   /* USER CODE BEGIN TIM2_MspPostInit 0 */
481
482   /* USER CODE END TIM2_MspPostInit 0 */
483
484     __HAL_RCC_GPIOB_CLK_ENABLE();
485     /**TIM2 GPIO Configuration
486     PB3     ------> TIM2_CH2
487     */
488     GPIO_InitStruct.Pin = GPIO_PIN_3;
489     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
490     GPIO_InitStruct.Pull = GPIO_NOPULL;
491     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
492     GPIO_InitStruct.Alternate = GPIO_AF1_TIM2;
493     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
494
495   /* USER CODE BEGIN TIM2_MspPostInit 1 */
496
```

```
497   /* USER CODE END TIM2_MspPostInit 1 */
498   }
499   else if(timHandle->Instance==TIM3)
500   {
501   /* USER CODE BEGIN TIM3_MspPostInit 0 */
502
503   /* USER CODE END TIM3_MspPostInit 0 */
504
505     __HAL_RCC_GPIOB_CLK_ENABLE();
506     /**TIM3 GPIO Configuration
507     PB5     ------> TIM3_CH2
508     */
509     GPIO_InitStruct.Pin = GPIO_PIN_5;
510     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
511     GPIO_InitStruct.Pull = GPIO_NOPULL;
512     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
513     GPIO_InitStruct.Alternate = GPIO_AF2_TIM3;
514     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
515
516   /* USER CODE BEGIN TIM3_MspPostInit 1 */
517
518   /* USER CODE END TIM3_MspPostInit 1 */
519   }
520   else if(timHandle->Instance==TIM4)
521   {
522   /* USER CODE BEGIN TIM4_MspPostInit 0 */
523
524   /* USER CODE END TIM4_MspPostInit 0 */
525
526     __HAL_RCC_GPIOB_CLK_ENABLE();
527     /**TIM4 GPIO Configuration
528     PB6     ------> TIM4_CH1
529     */
530     GPIO_InitStruct.Pin = GPIO_PIN_6;
531     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
532     GPIO_InitStruct.Pull = GPIO_NOPULL;
533     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
534     GPIO_InitStruct.Alternate = GPIO_AF2_TIM4;
535     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
536
537   /* USER CODE BEGIN TIM4_MspPostInit 1 */
538
539   /* USER CODE END TIM4_MspPostInit 1 */
540   }
541   else if(timHandle->Instance==TIM8)
542   {
543   /* USER CODE BEGIN TIM8_MspPostInit 0 */
544
545   /* USER CODE END TIM8_MspPostInit 0 */
546
547     __HAL_RCC_GPIOC_CLK_ENABLE();
548     /**TIM8 GPIO Configuration
549     PC7     ------> TIM8_CH2
550     */
551     GPIO_InitStruct.Pin = GPIO_PIN_7;
552     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
553     GPIO_InitStruct.Pull = GPIO_NOPULL;
554     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
555     GPIO_InitStruct.Alternate = GPIO_AF3_TIM8;
556     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
557
558   /* USER CODE BEGIN TIM8_MspPostInit 1 */
```

```
559
560   /* USER CODE END TIM8_MspPostInit 1 */
561   }
562   else if(timHandle->Instance==TIM10)
563   {
564   /* USER CODE BEGIN TIM10_MspPostInit 0 */
565
566   /* USER CODE END TIM10_MspPostInit 0 */
567
568     __HAL_RCC_GPIOB_CLK_ENABLE();
569     /**TIM10 GPIO Configuration
570     PB8     ------> TIM10_CH1
571     */
572     GPIO_InitStruct.Pin = GPIO_PIN_8;
573     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
574     GPIO_InitStruct.Pull = GPIO_NOPULL;
575     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
576     GPIO_InitStruct.Alternate = GPIO_AF3_TIM10;
577     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
578
579   /* USER CODE BEGIN TIM10_MspPostInit 1 */
580
581   /* USER CODE END TIM10_MspPostInit 1 */
582   }
583   else if(timHandle->Instance==TIM14)
584   {
585   /* USER CODE BEGIN TIM14_MspPostInit 0 */
586
587   /* USER CODE END TIM14_MspPostInit 0 */
588
589     __HAL_RCC_GPIOA_CLK_ENABLE();
590     /**TIM14 GPIO Configuration
591     PA7     ------> TIM14_CH1
592     */
593     GPIO_InitStruct.Pin = GPIO_PIN_7;
594     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
595     GPIO_InitStruct.Pull = GPIO_NOPULL;
596     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
597     GPIO_InitStruct.Alternate = GPIO_AF9_TIM14;
598     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
599
600   /* USER CODE BEGIN TIM14_MspPostInit 1 */
601
602   /* USER CODE END TIM14_MspPostInit 1 */
603   }
604
605 }
606
607 void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef* tim_baseHandle)
608 {
609
610   if(tim_baseHandle->Instance==TIM1)
611   {
612   /* USER CODE BEGIN TIM1_MspDeInit 0 */
613
614   /* USER CODE END TIM1_MspDeInit 0 */
615     /* Peripheral clock disable */
616     __HAL_RCC_TIM1_CLK_DISABLE();
617
618     /* TIM1 interrupt Deinit */
619   /* USER CODE BEGIN TIM1:TIM1_UP_TIM10_IRQn disable */
620     /**
```

```
621     * Uncomment the line below to disable the "TIM1_UP_TIM10_IRQn" interrupt
622     * Be aware, disabling shared interrupt may affect other IPs
623     */
624     /* HAL_NVIC_DisableIRQ(TIM1_UP_TIM10_IRQn); */
625   /* USER CODE END TIM1:TIM1_UP_TIM10_IRQn disable */
626
627   /* USER CODE BEGIN TIM1_MspDeInit 1 */
628
629   /* USER CODE END TIM1_MspDeInit 1 */
630   }
631   else if(tim_baseHandle->Instance==TIM2)
632   {
633   /* USER CODE BEGIN TIM2_MspDeInit 0 */
634
635   /* USER CODE END TIM2_MspDeInit 0 */
636     /* Peripheral clock disable */
637     __HAL_RCC_TIM2_CLK_DISABLE();
638
639     /* TIM2 interrupt Deinit */
640     HAL_NVIC_DisableIRQ(TIM2_IRQn);
641   /* USER CODE BEGIN TIM2_MspDeInit 1 */
642
643   /* USER CODE END TIM2_MspDeInit 1 */
644   }
645   else if(tim_baseHandle->Instance==TIM3)
646   {
647   /* USER CODE BEGIN TIM3_MspDeInit 0 */
648
649   /* USER CODE END TIM3_MspDeInit 0 */
650     /* Peripheral clock disable */
651     __HAL_RCC_TIM3_CLK_DISABLE();
652
653     /* TIM3 interrupt Deinit */
654     HAL_NVIC_DisableIRQ(TIM3_IRQn);
655   /* USER CODE BEGIN TIM3_MspDeInit 1 */
656
657   /* USER CODE END TIM3_MspDeInit 1 */
658   }
659   else if(tim_baseHandle->Instance==TIM4)
660   {
661   /* USER CODE BEGIN TIM4_MspDeInit 0 */
662
663   /* USER CODE END TIM4_MspDeInit 0 */
664     /* Peripheral clock disable */
665     __HAL_RCC_TIM4_CLK_DISABLE();
666
667     /* TIM4 interrupt Deinit */
668     HAL_NVIC_DisableIRQ(TIM4_IRQn);
669   /* USER CODE BEGIN TIM4_MspDeInit 1 */
670
671   /* USER CODE END TIM4_MspDeInit 1 */
672   }
673   else if(tim_baseHandle->Instance==TIM8)
674   {
675   /* USER CODE BEGIN TIM8_MspDeInit 0 */
676
677   /* USER CODE END TIM8_MspDeInit 0 */
678     /* Peripheral clock disable */
679     __HAL_RCC_TIM8_CLK_DISABLE();
680
681     /* TIM8 interrupt Deinit */
682     HAL_NVIC_DisableIRQ(TIM8_UP_TIM13_IRQn);
```

```
683   /* USER CODE BEGIN TIM8:TIM8_TRG_COM_TIM14_IRQn disable */
684     /**
685     * Uncomment the line below to disable the "TIM8_TRG_COM_TIM14_IRQn" interrupt
686     * Be aware, disabling shared interrupt may affect other IPs
687     */
688     /* HAL_NVIC_DisableIRQ(TIM8_TRG_COM_TIM14_IRQn); */
689   /* USER CODE END TIM8:TIM8_TRG_COM_TIM14_IRQn disable */
690
691   /* USER CODE BEGIN TIM8_MspDeInit 1 */
692
693   /* USER CODE END TIM8_MspDeInit 1 */
694   }
695   else if(tim_baseHandle->Instance==TIM10)
696   {
697   /* USER CODE BEGIN TIM10_MspDeInit 0 */
698
699   /* USER CODE END TIM10_MspDeInit 0 */
700     /* Peripheral clock disable */
701     __HAL_RCC_TIM10_CLK_DISABLE();
702
703     /* TIM10 interrupt Deinit */
704   /* USER CODE BEGIN TIM10:TIM1_UP_TIM10_IRQn disable */
705     /**
706     * Uncomment the line below to disable the "TIM1_UP_TIM10_IRQn" interrupt
707     * Be aware, disabling shared interrupt may affect other IPs
708     */
709     /* HAL_NVIC_DisableIRQ(TIM1_UP_TIM10_IRQn); */
710   /* USER CODE END TIM10:TIM1_UP_TIM10_IRQn disable */
711
712   /* USER CODE BEGIN TIM10_MspDeInit 1 */
713
714   /* USER CODE END TIM10_MspDeInit 1 */
715   }
716   else if(tim_baseHandle->Instance==TIM14)
717   {
718   /* USER CODE BEGIN TIM14_MspDeInit 0 */
719
720   /* USER CODE END TIM14_MspDeInit 0 */
721     /* Peripheral clock disable */
722     __HAL_RCC_TIM14_CLK_DISABLE();
723
724     /* TIM14 interrupt Deinit */
725   /* USER CODE BEGIN TIM14:TIM8_TRG_COM_TIM14_IRQn disable */
726     /**
727     * Uncomment the line below to disable the "TIM8_TRG_COM_TIM14_IRQn" interrupt
728     * Be aware, disabling shared interrupt may affect other IPs
729     */
730     /* HAL_NVIC_DisableIRQ(TIM8_TRG_COM_TIM14_IRQn); */
731   /* USER CODE END TIM14:TIM8_TRG_COM_TIM14_IRQn disable */
732
733   /* USER CODE BEGIN TIM14_MspDeInit 1 */
734
735   /* USER CODE END TIM14_MspDeInit 1 */
736   }
737 }
738
739 /* USER CODE BEGIN 1 */
740
741 /* USER CODE END 1 */
742
743 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
744
```

```c
/**
  ******************************************************************************
  * File Name          : TIM.h
  * Description        : This file provides code for the configuration
  *                      of the TIM instances.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __tim_H
#define __tim_H
#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

extern TIM_HandleTypeDef htim1;
extern TIM_HandleTypeDef htim2;
extern TIM_HandleTypeDef htim3;
extern TIM_HandleTypeDef htim4;
extern TIM_HandleTypeDef htim8;
extern TIM_HandleTypeDef htim10;
extern TIM_HandleTypeDef htim14;

/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

void MX_TIM1_Init(void);
void MX_TIM2_Init(void);
void MX_TIM3_Init(void);
void MX_TIM4_Init(void);
void MX_TIM8_Init(void);
void MX_TIM10_Init(void);
void MX_TIM14_Init(void);

void HAL_TIM_MspPostInit(TIM_HandleTypeDef *htim);

/* USER CODE BEGIN Prototypes */

/* USER CODE END Prototypes */

#ifdef __cplusplus
}
#endif
#endif /*__ tim_H */
```

```
63
64 /**
65   * @}
66   */
67
68 /**
69   * @}
70   */
71
72 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
73
```

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

IPAddress local_IP(192, 168, 1, 151);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);

char* ssid = "";
char* password = "";
WiFiServer server(80);

String header;

unsigned long currentTime = millis();
unsigned long previousTime = 0;
const long timeoutTime = 2000;

const int output1 = 16;
const int output2 = 5;
const int output3 = 4;
const int output4 = 0;

void setup()
{
    pinMode(output1, OUTPUT);
    pinMode(output2, OUTPUT);
    pinMode(output3, OUTPUT);
    pinMode(output4, OUTPUT);

    digitalWrite(output1, LOW);
    digitalWrite(output2, LOW);
    digitalWrite(output3, LOW);
    digitalWrite(output4, LOW);

    Serial.begin(115200);


    Serial.print("Connecting to ");
    Serial.println(ssid);

    if (!WiFi.config(local_IP, gateway, subnet))
    {
        Serial.println("STA failed to configure.");
    }

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
```

```
 57      server.begin();
 58
 59  }
 60
 61  void loop()
 62  {
 63      WiFiClient client = server.available();
 64
 65      if (client)
 66      {
 67          Serial.println("New Client.");
 68          String currentLine = "";
 69          currentTime = millis();
 70          previousTime = currentTime;
 71          while (client.connected() && currentTime - previousTime <=
               timeoutTime)
 72          {
 73              currentTime = millis();
 74              if (client.available())
 75              {
 76                  char c = client.read();
 77                  Serial.write(c);
 78                  Serial.println("\n");
 79                  header += c;
 80                  if (c == '0')
 81                  {
 82                      digitalWrite(output1, HIGH);
 83                      digitalWrite(output2, LOW);
 84                      digitalWrite(output3, LOW);
 85                      digitalWrite(output4, LOW);
 86                      delay(1000);
 87                      digitalWrite(output1, LOW);
 88                  }
 89                  else if (c == '1')
 90                  {
 91                      digitalWrite(output1, LOW);
 92                      digitalWrite(output2, HIGH);
 93                      digitalWrite(output3, LOW);
 94                      digitalWrite(output4, LOW);
 95                      delay(1000);
 96                      digitalWrite(output2, LOW);
 97                  }
 98                  else if (c == '2')
 99                  {
100                      digitalWrite(output1, LOW);
101                      digitalWrite(output2, LOW);
102                      digitalWrite(output3, HIGH);
103                      digitalWrite(output4, LOW);
104                      delay(1000);
105                      digitalWrite(output3, LOW);
106                  }
107                  else if (c == '3')
108                  {
109                      digitalWrite(output1, LOW);
110                      digitalWrite(output2, LOW);
111                      digitalWrite(output3, LOW);
```

```
112                     digitalWrite(output4, HIGH);
113                     delay(1000);
114                     digitalWrite(output4, LOW);
115                 }
116             }
117         }
118         header = "";
119         client.stop();
120         Serial.println("Client disconnected.");
121         Serial.println("");
122     }
123 }
```

```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM

#include "camera_pins.h"

IPAddress local_IP(192, 168, 1, 150);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);

const char* ssid = "";
const char* password = "";

void startCameraServer();

void setup()
{
    Serial.begin(115200);
    Serial.setDebugOutput(true);
    Serial.println();

    camera_config_t config;
    config.ledc_channel = LEDC_CHANNEL_0;
    config.ledc_timer = LEDC_TIMER_0;
    config.pin_d0 = Y2_GPIO_NUM;
    config.pin_d1 = Y3_GPIO_NUM;
    config.pin_d2 = Y4_GPIO_NUM;
    config.pin_d3 = Y5_GPIO_NUM;
    config.pin_d4 = Y6_GPIO_NUM;
    config.pin_d5 = Y7_GPIO_NUM;
    config.pin_d6 = Y8_GPIO_NUM;
    config.pin_d7 = Y9_GPIO_NUM;
    config.pin_xclk = XCLK_GPIO_NUM;
    config.pin_pclk = PCLK_GPIO_NUM;
    config.pin_vsync = VSYNC_GPIO_NUM;
    config.pin_href = HREF_GPIO_NUM;
    config.pin_sscb_sda = SIOD_GPIO_NUM;
    config.pin_sscb_scl = SIOC_GPIO_NUM;
    config.pin_pwdn = PWDN_GPIO_NUM;
    config.pin_reset = RESET_GPIO_NUM;
    config.xclk_freq_hz = 20000000;
```

```
57      config.pixel_format = PIXFORMAT_JPEG;
58
59      // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
60      //                      for larger pre-allocated frame buffer.
61      if (psramFound())
62      {
63          config.frame_size = FRAMESIZE_UXGA;
64          config.jpeg_quality = 10;
65          config.fb_count = 2;
66      }
67      else
68      {
69          config.frame_size = FRAMESIZE_SVGA;
70          config.jpeg_quality = 12;
71          config.fb_count = 1;
72      }
73
74  #if defined(CAMERA_MODEL_ESP_EYE)
75    pinMode(13, INPUT_PULLUP);
76    pinMode(14, INPUT_PULLUP);
77  #endif
78
79      // camera init
80      esp_err_t err = esp_camera_init(&config);
81      if (err != ESP_OK)
82      {
83          Serial.printf("Camera init failed with error 0x%x", err);
84          return;
85      }
86
87      sensor_t* s = esp_camera_sensor_get();
88      // initial sensors are flipped vertically and colors are a bit saturated
89      if (s->id.PID == OV3660_PID)
90      {
91          s->set_vflip(s, 1); // flip it back
92          s->set_brightness(s, 1); // up the brightness just a bit
93          s->set_saturation(s, -2); // lower the saturation
94      }
95      // drop down frame size for higher initial frame rate
96      s->set_framesize(s, FRAMESIZE_QVGA);
97
98  #if defined(CAMERA_MODEL_M5STACK_WIDE) || defined                              ↵
      (CAMERA_MODEL_M5STACK_ESP32CAM)
99    s->set_vflip(s, 1);
100   s->set_hmirror(s, 1);
101 #endif
102
103     if (!WiFi.config(local_IP, gateway, subnet))
104     {
105         Serial.println("STA failed to configure.");
106     }
107
108     WiFi.begin(ssid, password);
109
110     while (WiFi.status() != WL_CONNECTED)
111     {
```

```
112         delay(500);
113         Serial.print(".");
114     }
115     Serial.println("");
116     Serial.println("WiFi connected");
117 
118     startCameraServer();
119 
120     Serial.print("Camera Ready! Use 'http://");
121     Serial.print(WiFi.localIP());
122     Serial.println("' to connect");
123 }
124 
125 void loop()
126 {
127     delay(10000);
128 }
```

```
// Copyright 2015-2016 Espressif Systems (Shanghai) PTE LTD
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"

#include "fb_gfx.h"
#include "fd_forward.h"
#include "fr_forward.h"

#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

#define FACE_COLOR_WHITE  0x00FFFFFF
#define FACE_COLOR_BLACK  0x00000000
#define FACE_COLOR_RED    0x000000FF
#define FACE_COLOR_GREEN  0x0000FF00
#define FACE_COLOR_BLUE   0x00FF0000
#define FACE_COLOR_YELLOW (FACE_COLOR_RED | FACE_COLOR_GREEN)
#define FACE_COLOR_CYAN   (FACE_COLOR_BLUE | FACE_COLOR_GREEN)
#define FACE_COLOR_PURPLE (FACE_COLOR_BLUE | FACE_COLOR_RED)

typedef struct {
    size_t size; //number of values used for filtering
    size_t index; //current value index
    size_t count; //value count
    int sum;
    int* values; //array to be filled with values
} ra_filter_t;

typedef struct {
    httpd_req_t* req;
    size_t len;
} jpg_chunking_t;

#define PART_BOUNDARY "123456789000000000000987654321"
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";

```

```
55  static ra_filter_t ra_filter;
56  httpd_handle_t stream_httpd = NULL;
57  httpd_handle_t camera_httpd = NULL;
58
59  static mtmn_config_t mtmn_config = { 0 };
60  static int8_t detection_enabled = 0;
61  static int8_t recognition_enabled = 0;
62  static int8_t is_enrolling = 0;
63  static face_id_list id_list = { 0 };
64
65  static ra_filter_t* ra_filter_init(ra_filter_t* filter, size_t sample_size) {
66      memset(filter, 0, sizeof(ra_filter_t));
67
68      filter->values = (int*)malloc(sample_size * sizeof(int));
69      if (!filter->values) {
70          return NULL;
71      }
72      memset(filter->values, 0, sample_size * sizeof(int));
73
74      filter->size = sample_size;
75      return filter;
76  }
77
78  static int ra_filter_run(ra_filter_t* filter, int value) {
79      if (!filter->values) {
80          return value;
81      }
82      filter->sum -= filter->values[filter->index];
83      filter->values[filter->index] = value;
84      filter->sum += filter->values[filter->index];
85      filter->index++;
86      filter->index = filter->index % filter->size;
87      if (filter->count < filter->size) {
88          filter->count++;
89      }
90      return filter->sum / filter->count;
91  }
92
93  static void rgb_print(dl_matrix3du_t* image_matrix, uint32_t color, const
      char* str) {
94      fb_data_t fb;
95      fb.width = image_matrix->w;
96      fb.height = image_matrix->h;
97      fb.data = image_matrix->item;
98      fb.bytes_per_pixel = 3;
99      fb.format = FB_BGR888;
100     fb_gfx_print(&fb, (fb.width - (strlen(str) * 14)) / 2, 10, color, str);
101 }
102
103 static int rgb_printf(dl_matrix3du_t* image_matrix, uint32_t color, const
      char* format, ...) {
104     char loc_buf[64];
105     char* temp = loc_buf;
106     int len;
107     va_list arg;
108     va_list copy;
```

```
109     va_start(arg, format);
110     va_copy(copy, arg);
111     len = vsnprintf(loc_buf, sizeof(loc_buf), format, arg);
112     va_end(copy);
113     if (len >= sizeof(loc_buf)) {
114         temp = (char*)malloc(len + 1);
115         if (temp == NULL) {
116             return 0;
117         }
118     }
119     vsnprintf(temp, len + 1, format, arg);
120     va_end(arg);
121     rgb_print(image_matrix, color, temp);
122     if (len > 64) {
123         free(temp);
124     }
125     return len;
126 }
127 
128 static void draw_face_boxes(dl_matrix3du_t* image_matrix, box_array_t* boxes,
      int face_id) {
129     int x, y, w, h, i;
130     uint32_t color = FACE_COLOR_YELLOW;
131     if (face_id < 0) {
132         color = FACE_COLOR_RED;
133     }
134     else if (face_id > 0) {
135         color = FACE_COLOR_GREEN;
136     }
137     fb_data_t fb;
138     fb.width = image_matrix->w;
139     fb.height = image_matrix->h;
140     fb.data = image_matrix->item;
141     fb.bytes_per_pixel = 3;
142     fb.format = FB_BGR888;
143     for (i = 0; i < boxes->len; i++) {
144         // rectangle box
145         x = (int)boxes->box[i].box_p[0];
146         y = (int)boxes->box[i].box_p[1];
147         w = (int)boxes->box[i].box_p[2] - x + 1;
148         h = (int)boxes->box[i].box_p[3] - y + 1;
149         fb_gfx_drawFastHLine(&fb, x, y, w, color);
150         fb_gfx_drawFastHLine(&fb, x, y + h - 1, w, color);
151         fb_gfx_drawFastVLine(&fb, x, y, h, color);
152         fb_gfx_drawFastVLine(&fb, x + w - 1, y, h, color);
153 #if 0
154         // landmark
155         int x0, y0, j;
156         for (j = 0; j < 10; j += 2) {
157             x0 = (int)boxes->landmark[i].landmark_p[j];
158             y0 = (int)boxes->landmark[i].landmark_p[j + 1];
159             fb_gfx_fillRect(&fb, x0, y0, 3, 3, color);
160         }
161 #endif
162     }
163 }
```

```
164
165 static int run_face_recognition(dl_matrix3du_t* image_matrix, box_array_t*
      net_boxes) {
166     dl_matrix3du_t* aligned_face = NULL;
167     int matched_id = 0;
168
169     aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
170     if (!aligned_face) {
171         Serial.println("Could not allocate face recognition buffer");
172         return matched_id;
173     }
174     if (align_face(net_boxes, image_matrix, aligned_face) == ESP_OK) {
175         if (is_enrolling == 1) {
176             int8_t left_sample_face = enroll_face(&id_list, aligned_face);
177
178             if (left_sample_face == (ENROLL_CONFIRM_TIMES - 1)) {
179                 Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
180             }
181             Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail,
                  ENROLL_CONFIRM_TIMES - left_sample_face);
182             rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u] Sample[%u]",
                  id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
183             if (left_sample_face == 0) {
184                 is_enrolling = 0;
185                 Serial.printf("Enrolled Face ID: %d\n", id_list.tail);
186             }
187         }
188         else {
189             matched_id = recognize_face(&id_list, aligned_face);
190             if (matched_id >= 0) {
191                 Serial.printf("Match Face ID: %u\n", matched_id);
192                 rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject %u",
                      matched_id);
193             }
194             else {
195                 Serial.println("No Match Found");
196                 rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
197                 matched_id = -1;
198             }
199         }
200     }
201     else {
202         Serial.println("Face Not Aligned");
203         //rgb_print(image_matrix, FACE_COLOR_YELLOW, "Human Detected");
204     }
205
206     dl_matrix3du_free(aligned_face);
207     return matched_id;
208 }
209
210 static size_t jpg_encode_stream(void* arg, size_t index, const void* data,
      size_t len) {
211     jpg_chunking_t* j = (jpg_chunking_t*)arg;
212     if (!index) {
213         j->len = 0;
214     }
```

```
215     if (httpd_resp_send_chunk(j->req, (const char*)data, len) != ESP_OK) {
216         return 0;
217     }
218     j->len += len;
219     return len;
220 }
221
222 static esp_err_t capture_handler(httpd_req_t* req) {
223     camera_fb_t* fb = NULL;
224     esp_err_t res = ESP_OK;
225     int64_t fr_start = esp_timer_get_time();
226
227     fb = esp_camera_fb_get();
228     if (!fb) {
229         Serial.println("Camera capture failed");
230         httpd_resp_send_500(req);
231         return ESP_FAIL;
232     }
233
234     httpd_resp_set_type(req, "image/jpeg");
235     httpd_resp_set_hdr(req, "Content-Disposition", "inline;
        filename=capture.jpg");
236     httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
237
238     size_t out_len, out_width, out_height;
239     uint8_t* out_buf;
240     bool s;
241     bool detected = false;
242     int face_id = 0;
243     if (!detection_enabled || fb->width > 400) {
244         size_t fb_len = 0;
245         if (fb->format == PIXFORMAT_JPEG) {
246             fb_len = fb->len;
247             res = httpd_resp_send(req, (const char*)fb->buf, fb->len);
248         }
249         else {
250             jpg_chunking_t jchunk = { req, 0 };
251             res = frame2jpg_cb(fb, 80, jpg_encode_stream, &jchunk) ? ESP_OK :
                ESP_FAIL;
252             httpd_resp_send_chunk(req, NULL, 0);
253             fb_len = jchunk.len;
254         }
255         esp_camera_fb_return(fb);
256         int64_t fr_end = esp_timer_get_time();
257         Serial.printf("JPG: %uB %ums\n", (uint32_t)(fb_len), (uint32_t)
            ((fr_end - fr_start) / 1000));
258         return res;
259     }
260
261     dl_matrix3du_t* image_matrix = dl_matrix3du_alloc(1, fb->width, fb-
        >height, 3);
262     if (!image_matrix) {
263         esp_camera_fb_return(fb);
264         Serial.println("dl_matrix3du_alloc failed");
265         httpd_resp_send_500(req);
266         return ESP_FAIL;
```

```
267     }
268
269     out_buf = image_matrix->item;
270     out_len = fb->width * fb->height * 3;
271     out_width = fb->width;
272     out_height = fb->height;
273
274     s = fmt2rgb888(fb->buf, fb->len, fb->format, out_buf);
275     esp_camera_fb_return(fb);
276     if (!s) {
277         dl_matrix3du_free(image_matrix);
278         Serial.println("to rgb888 failed");
279         httpd_resp_send_500(req);
280         return ESP_FAIL;
281     }
282
283     box_array_t* net_boxes = face_detect(image_matrix, &mtmn_config);
284
285     if (net_boxes) {
286         detected = true;
287         if (recognition_enabled) {
288             face_id = run_face_recognition(image_matrix, net_boxes);
289         }
290         draw_face_boxes(image_matrix, net_boxes, face_id);
291         free(net_boxes->score);
292         free(net_boxes->box);
293         free(net_boxes->landmark);
294         free(net_boxes);
295     }
296
297     jpg_chunking_t jchunk = { req, 0 };
298     s = fmt2jpg_cb(out_buf, out_len, out_width, out_height, PIXFORMAT_RGB888, ↵
          90, jpg_encode_stream, &jchunk);
299     dl_matrix3du_free(image_matrix);
300     if (!s) {
301         Serial.println("JPEG compression failed");
302         return ESP_FAIL;
303     }
304
305     int64_t fr_end = esp_timer_get_time();
306     Serial.printf("FACE: %uB %ums %s%d\n", (uint32_t)(jchunk.len), (uint32_t) ↵
          ((fr_end - fr_start) / 1000), detected ? "DETECTED " : "", face_id);
307     return res;
308 }
309
310 static esp_err_t stream_handler(httpd_req_t* req) {
311     camera_fb_t* fb = NULL;
312     esp_err_t res = ESP_OK;
313     size_t _jpg_buf_len = 0;
314     uint8_t* _jpg_buf = NULL;
315     char* part_buf[64];
316     dl_matrix3du_t* image_matrix = NULL;
317     bool detected = false;
318     int face_id = 0;
319     int64_t fr_start = 0;
320     int64_t fr_ready = 0;
```

```
321     int64_t fr_face = 0;
322     int64_t fr_recognize = 0;
323     int64_t fr_encode = 0;
324
325     static int64_t last_frame = 0;
326     if (!last_frame) {
327         last_frame = esp_timer_get_time();
328     }
329
330     res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
331     if (res != ESP_OK) {
332         return res;
333     }
334
335     httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
336
337     while (true) {
338         detected = false;
339         face_id = 0;
340         fb = esp_camera_fb_get();
341         if (!fb) {
342             Serial.println("Camera capture failed");
343             res = ESP_FAIL;
344         }
345         else {
346             fr_start = esp_timer_get_time();
347             fr_ready = fr_start;
348             fr_face = fr_start;
349             fr_encode = fr_start;
350             fr_recognize = fr_start;
351             if (!detection_enabled || fb->width > 400) {
352                 if (fb->format != PIXFORMAT_JPEG) {
353                     bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf,
                        &_jpg_buf_len);
354                     esp_camera_fb_return(fb);
355                     fb = NULL;
356                     if (!jpeg_converted) {
357                         Serial.println("JPEG compression failed");
358                         res = ESP_FAIL;
359                     }
360                 }
361                 else {
362                     _jpg_buf_len = fb->len;
363                     _jpg_buf = fb->buf;
364                 }
365             }
366             else {
367
368                 image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height,
                      3);
369
370                 if (!image_matrix) {
371                     Serial.println("dl_matrix3du_alloc failed");
372                     res = ESP_FAIL;
373                 }
374                 else {
```

```
375                     if (!fmt2rgb888(fb->buf, fb->len, fb->format,
                        image_matrix->item)) {
376                         Serial.println("fmt2rgb888 failed");
377                         res = ESP_FAIL;
378                     }
379                     else {
380                         fr_ready = esp_timer_get_time();
381                         box_array_t* net_boxes = NULL;
382                         if (detection_enabled) {
383                             net_boxes = face_detect(image_matrix,
                        &mtmn_config);
384                         }
385                         fr_face = esp_timer_get_time();
386                         fr_recognize = fr_face;
387                         if (net_boxes || fb->format != PIXFORMAT_JPEG) {
388                             if (net_boxes) {
389                                 detected = true;
390                                 if (recognition_enabled) {
391                                     face_id = run_face_recognition
                        (image_matrix, net_boxes);
392                                 }
393                                 fr_recognize = esp_timer_get_time();
394                                 draw_face_boxes(image_matrix, net_boxes,
                        face_id);
395                                 free(net_boxes->score);
396                                 free(net_boxes->box);
397                                 free(net_boxes->landmark);
398                                 free(net_boxes);
399                             }
400                             if (!fmt2jpg(image_matrix->item, fb->width * fb-
                        >height * 3, fb->width, fb->height, PIXFORMAT_RGB888, 90,
                        &_jpg_buf, &_jpg_buf_len)) {
401                                 Serial.println("fmt2jpg failed");
402                                 res = ESP_FAIL;
403                             }
404                             esp_camera_fb_return(fb);
405                             fb = NULL;
406                         }
407                         else {
408                             _jpg_buf = fb->buf;
409                             _jpg_buf_len = fb->len;
410                         }
411                         fr_encode = esp_timer_get_time();
412                     }
413                     dl_matrix3du_free(image_matrix);
414                 }
415             }
416         }
417         if (res == ESP_OK) {
418             res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen
                  (_STREAM_BOUNDARY));
419         }
420         if (res == ESP_OK) {
421             size_t hlen = snprintf((char*)part_buf, 64, _STREAM_PART,
                  _jpg_buf_len);
422             res = httpd_resp_send_chunk(req, (const char*)part_buf, hlen);
```

```
423         }
424         if (res == ESP_OK) {
425             res = httpd_resp_send_chunk(req, (const char*)_jpg_buf,
                  _jpg_buf_len);
426         }
427         if (fb) {
428             esp_camera_fb_return(fb);
429             fb = NULL;
430             _jpg_buf = NULL;
431         }
432         else if (_jpg_buf) {
433             free(_jpg_buf);
434             _jpg_buf = NULL;
435         }
436         if (res != ESP_OK) {
437             break;
438         }
439         int64_t fr_end = esp_timer_get_time();
440
441         int64_t ready_time = (fr_ready - fr_start) / 1000;
442         int64_t face_time = (fr_face - fr_ready) / 1000;
443         int64_t recognize_time = (fr_recognize - fr_face) / 1000;
444         int64_t encode_time = (fr_encode - fr_recognize) / 1000;
445         int64_t process_time = (fr_encode - fr_start) / 1000;
446
447         int64_t frame_time = fr_end - last_frame;
448         last_frame = fr_end;
449         frame_time /= 1000;
450         uint32_t avg_frame_time = ra_filter_run(&ra_filter, frame_time);
451         Serial.printf("MJPG: %uB %ums (%.1ffps), AVG: %ums (%.1ffps), %u+%u+%u
              +%u=%u %s%d\n",
452             (uint32_t)(_jpg_buf_len),
453             (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time,
454             avg_frame_time, 1000.0 / avg_frame_time,
455             (uint32_t)ready_time, (uint32_t)face_time, (uint32_t)
                  recognize_time, (uint32_t)encode_time, (uint32_t)process_time,
456             (detected) ? "DETECTED " : "", face_id
457         );
458     }
459
460     last_frame = 0;
461     return res;
462 }
463
464 static esp_err_t cmd_handler(httpd_req_t* req) {
465     char* buf;
466     size_t buf_len;
467     char variable[32] = { 0, };
468     char value[32] = { 0, };
469
470     buf_len = httpd_req_get_url_query_len(req) + 1;
471     if (buf_len > 1) {
472         buf = (char*)malloc(buf_len);
473         if (!buf) {
474             httpd_resp_send_500(req);
475             return ESP_FAIL;
```

```
476         }
477         if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
478             if (httpd_query_key_value(buf, "var", variable, sizeof(variable))
                == ESP_OK &&
479                 httpd_query_key_value(buf, "val", value, sizeof(value)) ==
                    ESP_OK) {
480             }
481             else {
482                 free(buf);
483                 httpd_resp_send_404(req);
484                 return ESP_FAIL;
485             }
486         }
487         else {
488             free(buf);
489             httpd_resp_send_404(req);
490             return ESP_FAIL;
491         }
492         free(buf);
493     }
494     else {
495         httpd_resp_send_404(req);
496         return ESP_FAIL;
497     }
498
499     int val = atoi(value);
500     sensor_t* s = esp_camera_sensor_get();
501     int res = 0;
502
503     if (!strcmp(variable, "framesize")) {
504         if (s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s,
              (framesize_t)val);
505     }
506     else if (!strcmp(variable, "quality")) res = s->set_quality(s, val);
507     else if (!strcmp(variable, "contrast")) res = s->set_contrast(s, val);
508     else if (!strcmp(variable, "brightness")) res = s->set_brightness(s, val);
509     else if (!strcmp(variable, "saturation")) res = s->set_saturation(s, val);
510     else if (!strcmp(variable, "gainceiling")) res = s->set_gainceiling(s,
          (gainceiling_t)val);
511     else if (!strcmp(variable, "colorbar")) res = s->set_colorbar(s, val);
512     else if (!strcmp(variable, "awb")) res = s->set_whitebal(s, val);
513     else if (!strcmp(variable, "agc")) res = s->set_gain_ctrl(s, val);
514     else if (!strcmp(variable, "aec")) res = s->set_exposure_ctrl(s, val);
515     else if (!strcmp(variable, "hmirror")) res = s->set_hmirror(s, val);
516     else if (!strcmp(variable, "vflip")) res = s->set_vflip(s, val);
517     else if (!strcmp(variable, "awb_gain")) res = s->set_awb_gain(s, val);
518     else if (!strcmp(variable, "agc_gain")) res = s->set_agc_gain(s, val);
519     else if (!strcmp(variable, "aec_value")) res = s->set_aec_value(s, val);
520     else if (!strcmp(variable, "aec2")) res = s->set_aec2(s, val);
521     else if (!strcmp(variable, "dcw")) res = s->set_dcw(s, val);
522     else if (!strcmp(variable, "bpc")) res = s->set_bpc(s, val);
523     else if (!strcmp(variable, "wpc")) res = s->set_wpc(s, val);
524     else if (!strcmp(variable, "raw_gma")) res = s->set_raw_gma(s, val);
525     else if (!strcmp(variable, "lenc")) res = s->set_lenc(s, val);
526     else if (!strcmp(variable, "special_effect")) res = s->set_special_effect
          (s, val);
```

```
527     else if (!strcmp(variable, "wb_mode")) res = s->set_wb_mode(s, val);
528     else if (!strcmp(variable, "ae_level")) res = s->set_ae_level(s, val);
529     else if (!strcmp(variable, "face_detect")) {
530         detection_enabled = val;
531         if (!detection_enabled) {
532             recognition_enabled = 0;
533         }
534     }
535     else if (!strcmp(variable, "face_enroll")) is_enrolling = val;
536     else if (!strcmp(variable, "face_recognize")) {
537         recognition_enabled = val;
538         if (recognition_enabled) {
539             detection_enabled = val;
540         }
541     }
542     else {
543         res = -1;
544     }
545 
546     if (res) {
547         return httpd_resp_send_500(req);
548     }
549 
550     httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
551     return httpd_resp_send(req, NULL, 0);
552 }
553 
554 static esp_err_t status_handler(httpd_req_t* req) {
555     static char json_response[1024];
556 
557     sensor_t* s = esp_camera_sensor_get();
558     char* p = json_response;
559     *p++ = '{';
560 
561     p += sprintf(p, "\"framesize\":%u,", s->status.framesize);
562     p += sprintf(p, "\"quality\":%u,", s->status.quality);
563     p += sprintf(p, "\"brightness\":%d,", s->status.brightness);
564     p += sprintf(p, "\"contrast\":%d,", s->status.contrast);
565     p += sprintf(p, "\"saturation\":%d,", s->status.saturation);
566     p += sprintf(p, "\"sharpness\":%d,", s->status.sharpness);
567     p += sprintf(p, "\"special_effect\":%u,", s->status.special_effect);
568     p += sprintf(p, "\"wb_mode\":%u,", s->status.wb_mode);
569     p += sprintf(p, "\"awb\":%u,", s->status.awb);
570     p += sprintf(p, "\"awb_gain\":%u,", s->status.awb_gain);
571     p += sprintf(p, "\"aec\":%u,", s->status.aec);
572     p += sprintf(p, "\"aec2\":%u,", s->status.aec2);
573     p += sprintf(p, "\"ae_level\":%d,", s->status.ae_level);
574     p += sprintf(p, "\"aec_value\":%u,", s->status.aec_value);
575     p += sprintf(p, "\"agc\":%u,", s->status.agc);
576     p += sprintf(p, "\"agc_gain\":%u,", s->status.agc_gain);
577     p += sprintf(p, "\"gainceiling\":%u,", s->status.gainceiling);
578     p += sprintf(p, "\"bpc\":%u,", s->status.bpc);
579     p += sprintf(p, "\"wpc\":%u,", s->status.wpc);
580     p += sprintf(p, "\"raw_gma\":%u,", s->status.raw_gma);
581     p += sprintf(p, "\"lenc\":%u,", s->status.lenc);
582     p += sprintf(p, "\"vflip\":%u,", s->status.vflip);
```

```
583     p += sprintf(p, "\"hmirror\":%u,", s->status.hmirror);
584     p += sprintf(p, "\"dcw\":%u,", s->status.dcw);
585     p += sprintf(p, "\"colorbar\":%u,", s->status.colorbar);
586     p += sprintf(p, "\"face_detect\":%u,", detection_enabled);
587     p += sprintf(p, "\"face_enroll\":%u,", is_enrolling);
588     p += sprintf(p, "\"face_recognize\":%u", recognition_enabled);
589     *p++ = '}';
590     *p++ = 0;
591     httpd_resp_set_type(req, "application/json");
592     httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
593     return httpd_resp_send(req, json_response, strlen(json_response));
594 }
595 
596 static esp_err_t index_handler(httpd_req_t* req) {
597     httpd_resp_set_type(req, "text/html");
598     httpd_resp_set_hdr(req, "Content-Encoding", "gzip");
599     sensor_t* s = esp_camera_sensor_get();
600     if (s->id.PID == OV3660_PID) {
601         return httpd_resp_send(req, (const char*)index_ov3660_html_gz,
            index_ov3660_html_gz_len);
602     }
603     return httpd_resp_send(req, (const char*)index_ov2640_html_gz,
      index_ov2640_html_gz_len);
604 }
605 
606 void startCameraServer() {
607     httpd_config_t config = HTTPD_DEFAULT_CONFIG();
608 
609     httpd_uri_t index_uri = {
610         .uri = "/",
611         .method = HTTP_GET,
612         .handler = index_handler,
613         .user_ctx = NULL
614     };
615 
616     httpd_uri_t status_uri = {
617         .uri = "/status",
618         .method = HTTP_GET,
619         .handler = status_handler,
620         .user_ctx = NULL
621     };
622 
623     httpd_uri_t cmd_uri = {
624         .uri = "/control",
625         .method = HTTP_GET,
626         .handler = cmd_handler,
627         .user_ctx = NULL
628     };
629 
630     httpd_uri_t capture_uri = {
631         .uri = "/capture",
632         .method = HTTP_GET,
633         .handler = capture_handler,
634         .user_ctx = NULL
635     };
636 
```

```
637     httpd_uri_t stream_uri = {
638          .uri = "/stream",
639          .method = HTTP_GET,
640          .handler = stream_handler,
641          .user_ctx = NULL
642     };
643
644
645     ra_filter_init(&ra_filter, 20);
646
647     mtmn_config.type = FAST;
648     mtmn_config.min_face = 80;
649     mtmn_config.pyramid = 0.707;
650     mtmn_config.pyramid_times = 4;
651     mtmn_config.p_threshold.score = 0.6;
652     mtmn_config.p_threshold.nms = 0.7;
653     mtmn_config.p_threshold.candidate_number = 20;
654     mtmn_config.r_threshold.score = 0.7;
655     mtmn_config.r_threshold.nms = 0.7;
656     mtmn_config.r_threshold.candidate_number = 10;
657     mtmn_config.o_threshold.score = 0.7;
658     mtmn_config.o_threshold.nms = 0.7;
659     mtmn_config.o_threshold.candidate_number = 1;
660
661     face_id_init(&id_list, FACE_ID_SAVE_NUMBER, ENROLL_CONFIRM_TIMES);
662
663     Serial.printf("Starting web server on port: '%d'\n", config.server_port);
664     if (httpd_start(&camera_httpd, &config) == ESP_OK) {
665         httpd_register_uri_handler(camera_httpd, &index_uri);
666         httpd_register_uri_handler(camera_httpd, &cmd_uri);
667         httpd_register_uri_handler(camera_httpd, &status_uri);
668         httpd_register_uri_handler(camera_httpd, &capture_uri);
669     }
670
671     config.server_port += 1;
672     config.ctrl_port += 1;
673     Serial.printf("Starting stream server on port: '%d'\n",
        config.server_port);
674     if (httpd_start(&stream_httpd, &config) == ESP_OK) {
675         httpd_register_uri_handler(stream_httpd, &stream_uri);
676     }
677 }
678
```

Spis rysunków