

Lampiran

```
1  #include "esp_camera.h"
2  #include <Arduino.h>
3  #include <WiFi.h>
4  #include <AsyncTCP.h>
5  #include <ESPAsyncWebServer.h>
6  #include <iostream>
7  #include <sstream>
8
9  struct MOTOR_PINS
10 {
11     int pinEn;
12     int pinIN1;
13     int pinIN2;
14 };
15
16 std::vector<MOTOR_PINS> motorPins =
17 {
18     {12, 13, 15}, //RIGHT_MOTOR Pins (EnA, IN1, IN2)
19     {12, 14, 2},  //LEFT_MOTOR Pins (EnB, IN3, IN4)
20 };
21 #define LIGHT_PIN 4
22
23 #define UP 1
24 #define DOWN 2
25 #define LEFT 3
26 #define RIGHT 4
27 #define STOP 0
28
29 #define RIGHT_MOTOR 0
30 #define LEFT_MOTOR 1
31
32 #define FORWARD 1
33 #define BACKWARD -1
34
35 const int PwmFreq = 1000; /* 1 KHz */
36 const int PwmResolution = 8;
37 const int PwmSpeedChannel = 2;
38 const int PwmLightChannel = 3;
39
40 //Camera related constants
41 #define PWDN_GPIO_NUM 32
42 #define RESET_GPIO_NUM -1
43 #define XCLK_GPIO_NUM 0
44 #define SIOD_GPIO_NUM 26
45 #define SIOC_GPIO_NUM 27
46 #define Y9_GPIO_NUM 35
47 #define Y8_GPIO_NUM 34
48 #define Y7_GPIO_NUM 39
49 #define Y6_GPIO_NUM 36
50 #define Y5_GPIO_NUM 21
51 #define Y4_GPIO_NUM 19
52 #define Y3_GPIO_NUM 18
53 #define Y2_GPIO_NUM 5
54 #define VSYNC_GPIO_NUM 25
55 #define HREF_GPIO_NUM 23
56 #define PCLK_GPIO_NUM 22
57
58 const char* ssid = "camera WIFI";
59 const char* password = "123456789";
60
61 AsyncWebServer server(80);
62 AsyncWebsocket wsCamera("/Camera");
63 AsyncWebsocket wsCarInput("/CarInput");
64 uint32_t cameraClientId = 0;
65
66 const char* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(
67 <!DOCTYPE html>
68 <html>
69 <head>
70 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
71 <style>
72 .arrows {
73     font-size:40px;
74     color:red;
75 }
76 td.button {
77     background-color:black;
78     border-radius:25%;
79     box-shadow: 5px 5px #888888;
80 }
81 td.button:active {
82     transform: translate(5px,5px);
83     box-shadow: none;
84 }
85 )"
```

```

85
86 .noselect {
87     -webkit-touch-callout: none; /* iOS Safari */
88     -webkit-user-select: none; /* Safari */
89     -khtml-user-select: none; /* Konqueror HTML */
90     -moz-user-select: none; /* Firefox */
91     -ms-user-select: none; /* Internet Explorer/Edge */
92     user-select: none; /* Non-prefixed version, currently
93         supported by Chrome and Opera */
94 }
95
96 .slidecontainer {
97     width: 100%;
98 }
99
100 .slider {
101     -webkit-appearance: none;
102     width: 100%;
103     height: 15px;
104     border-radius: 5px;
105     background: #d3d3d3;
106     outline: none;
107     opacity: 0.7;
108     -webkit-transition: .2s;
109     transition: opacity .2s;
110 }
111
112 .slider:hover {
113     opacity: 1;
114 }
115
116 .slider::-webkit-slider-thumb {
117     -webkit-appearance: none;
118     appearance: none;
119     width: 25px;
120     height: 25px;
121     border-radius: 50%;
122     background: red;
123     cursor: pointer;
124 }
125
126 .slider::-moz-range-thumb {
127     width: 25px;
128     height: 25px;
129     border-radius: 50%;
130     background: red;
131     cursor: pointer;
132 }
133
134 </style>
135
136 </head>
137 <body class="noselect" align="center" style="background-color:white">
138
139 <!--h2 style="color: teal;text-align:center;">Wi-Fi Camera &#128663; Control</h2-->
140
141 <table id="mainTable" style="width:400px;margin:auto;table-layout:fixed" CELLSPACING=10>
142 <tr>
143     <img id="cameraImage" src="" style="width:400px;height:250px"></td>
144 </tr>
145 <tr>
146     <td></td>
147     <td class="button" onclick="sendButtonInput('MoveCar','1') onclickend='sendButtonInput('MoveCar','0')"><span class="arrows" >&#8679;</span></td>
148     <td></td>
149 </tr>
150 <tr>
151     <td class="button" onclick="sendButtonInput('MoveCar','3') onclickend='sendButtonInput('MoveCar','0')"><span class="arrows" >&#8678;</span></td>
152     <td class="button"></td>
153     <td class="button" onclick="sendButtonInput('MoveCar','4') onclickend='sendButtonInput('MoveCar','0')"><span class="arrows" >&#8680;</span></td>
154 </tr>
155 <tr>
156     <td></td>
157     <td class="button" onclick="sendButtonInput('MoveCar','2') onclickend='sendButtonInput('MoveCar','0')"><span class="arrows" >&#8681;</span></td>
158     <td></td>
159 </tr>
160 <tr><tr>
161 <tr>
162     <td style="text-align:left"><b>Speed:</b></td>
163     <td colspan=2>
164         <div class="slidecontainer">
165             <input type="range" min="0" max="255" value="150" class="slider" id="Speed" oninput="sendButtonInput('Speed',value)">
166         </div>
167     </td>
168 </tr>
169 <tr>
170     <td style="text-align:left"><b>Light:</b></td>
171     <td colspan=2>
172         <div class="slidecontainer">
173             <input type="range" min="0" max="255" value="0" class="slider" id="Light" oninput="sendButtonInput('Light',value)">
174         </div>
175     </td>
176 </tr>
177 </table>
178
179 <script>
180     var websocketCameraUrl = "ws:///" + window.location.hostname + "/Camera";
181     var websocketCarInputUrl = "ws:///" + window.location.hostname + "/CarInput";
182     var websocketCamera;
183     var websocketCarInput;

```

```

185     function initCameraWebSocket()
186     {
187         websocketCamera = new WebSocket(webSocketCameraUrl);
188         websocketCamera.binaryType = 'blob';
189         websocketCamera.onopen = function(event){};
190         websocketCamera.onclose = function(event){setTimeout(initCameraWebSocket, 2000)};
191         websocketCamera.onmessage = function(event)
192         {
193             var imageId = document.getElementById("cameraImage");
194             imageId.src = URL.createObjectURL(event.data);
195         };
196     }
197
198     function initCarInputWebSocket()
199     {
200         websocketCarInput = new WebSocket(webSocketCarInputUrl);
201         websocketCarInput.onopen = function(event)
202         {
203             var speedButton = document.getElementById("Speed");
204             sendButtonInput("Speed", speedButton.value);
205             var lightButton = document.getElementById("Light");
206             sendButtonInput("Light", lightButton.value);
207         };
208         websocketCarInput.onclose = function(event){setTimeout(initCarInputWebSocket, 2000)};
209         websocketCarInput.onmessage = function(event){};
210     }
211
212     function initWebSocket()
213     {
214         initCameraWebSocket ();
215         initCarInputWebSocket();
216     }
217
218     function sendButtonInput(key, value)
219     {
220         var data = key + "," + value;
221         websocketCarInput.send(data);
222     }
223
224     window.onload = initWebSocket;
225     document.getElementById("mainTable").addEventListener("touchend", function(event){
226         event.preventDefault()
227     });
228 </script>
229 </body>
230 </html>
231 )HTMLHOMEPAGE";
232
233
234 void rotateMotor(int motorNumber, int motorDirection)
235 {
236     if (motorDirection == FORWARD)
237     {
238         digitalWrite(motorPins[motorNumber].pinIN1, HIGH);
239         digitalWrite(motorPins[motorNumber].pinIN2, LOW);
240     }
241     else if (motorDirection == BACKWARD)
242     {
243         digitalWrite(motorPins[motorNumber].pinIN1, LOW);
244         digitalWrite(motorPins[motorNumber].pinIN2, HIGH);
245     }
246     else
247     {
248         digitalWrite(motorPins[motorNumber].pinIN1, LOW);
249         digitalWrite(motorPins[motorNumber].pinIN2, LOW);
250     }
251 }
252
253 void moveCar(int inputValue)
254 {
255     Serial.printf("Got value as %d\n", inputValue);
256     switch(inputValue)
257     {
258     case UP:
259         rotateMotor(RIGHT_MOTOR, FORWARD);
260         rotateMotor(LEFT_MOTOR, FORWARD);
261         break;
262     case DOWN:
263         rotateMotor(RIGHT_MOTOR, BACKWARD);
264         rotateMotor(LEFT_MOTOR, BACKWARD);
265         break;
266     case LEFT:
267         rotateMotor(RIGHT_MOTOR, FORWARD);
268         rotateMotor(LEFT_MOTOR, BACKWARD);
269         break;
270     case RIGHT:
271         rotateMotor(RIGHT_MOTOR, BACKWARD);
272         rotateMotor(LEFT_MOTOR, FORWARD);
273         break;
274     }
275 }

```

```

278 |
279 | case STOP:
280 |     rotateMotor(RIGHT_MOTOR, STOP);
281 |     rotateMotor(LEFT_MOTOR, STOP);
282 |     break;
283 |
284 | default:
285 |     rotateMotor(RIGHT_MOTOR, STOP);
286 |     rotateMotor(LEFT_MOTOR, STOP);
287 |     break;
288 | }
289 | }
290 |
291 | void handleRoot(AsyncWebServerRequest *request)
292 | {
293 |     request->send_P(200, "text/html", htmlHomePage);
294 | }
295 |
296 | void handleNotFound(AsyncWebServerRequest *request)
297 | {
298 |     request->send(404, "text/plain", "File Not Found");
299 | }
300 |
301 | void onCarInputWebSocketEvent(AsyncWebSocket *server,
302 |                               AsyncWebSocketClient *client,
303 |                               AwsEventType type,
304 |                               void *arg,
305 |                               uint8_t *data,
306 |                               size_t len)
307 | {
308 |     switch (type)
309 |     {
310 |     case WS_EVT_CONNECT:
311 |         Serial.printf("WebSocket client #%u connected from %s\n", client->id(), client->remoteIP().toString().c_str());
312 |         break;
313 |     case WS_EVT_DISCONNECT:
314 |         Serial.printf("WebSocket client #%u disconnected\n", client->id());
315 |         moveCar(0);
316 |         ledcWrite(PWMLightChannel, 0);
317 |         break;
318 |     case WS_EVT_DATA:
319 |         AwsFrameInfo *info;
320 |         info = (AwsFrameInfo*)arg;
321 |         if (info->final && info->index == 0 && info->len == len && info->opcode == WS_TEXT)
322 |         {
323 |             std::string myData = "";
324 |             myData.assign((char *)data, len);
325 |             std::istringstream ss(myData);
326 |             std::string key, value;
327 |             std::getline(ss, key, ',');
328 |             std::getline(ss, value, ',');
329 |             Serial.printf("Key [%s] Value[%s]\n", key.c_str(), value.c_str());
330 |             int valueInt = atoi(value.c_str());
331 |             if (key == "MoveCar")
332 |             {
333 |                 moveCar(valueInt);
334 |             }
335 |             else if (key == "Speed")
336 |             {
337 |                 ledcWrite(PWMSpeedChannel, valueInt);
338 |             }
339 |             else if (key == "Light")
340 |             {
341 |                 ledcWrite(PWMLightChannel, valueInt);
342 |             }
343 |         }
344 |         break;
345 |     case WS_EVT_PONG:
346 |     case WS_EVT_ERROR:
347 |         break;
348 |     default:
349 |         break;
350 |     }
351 | }
352 |
353 | void onCameraWebSocketEvent(AsyncWebSocket *server,
354 |                             AsyncWebSocketClient *client,
355 |                             AwsEventType type,
356 |                             void *arg,
357 |                             uint8_t *data,
358 |                             size_t len)
359 | {
360 |     switch (type)
361 |     {
362 |     case WS_EVT_CONNECT:
363 |         Serial.printf("WebSocket client #%u connected from %s\n", client->id(), client->remoteIP().toString().c_str());
364 |         cameraClientId = client->id();
365 |         break;
366 |     case WS_EVT_DISCONNECT:
367 |         Serial.printf("WebSocket client #%u disconnected\n", client->id());
368 |         cameraClientId = 0;
369 |         break;
370 |     case WS_EVT_DATA:
371 |         break;
372 |     case WS_EVT_PONG:
373 |     case WS_EVT_ERROR:
374 |         break;

```

```

376     || break;
377 }
378 }
379
380 void setupCamera()
381 {
382     camera_config_t config;
383     config.ledc_channel = LEDC_CHANNEL_0;
384     config.ledc_timer = LEDC_TIMER_0;
385     config.pin_d0 = Y2_GPIO_NUM;
386     config.pin_d1 = Y3_GPIO_NUM;
387     config.pin_d2 = Y4_GPIO_NUM;
388     config.pin_d3 = Y5_GPIO_NUM;
389     config.pin_d4 = Y6_GPIO_NUM;
390     config.pin_d5 = Y7_GPIO_NUM;
391     config.pin_d6 = Y8_GPIO_NUM;
392     config.pin_d7 = Y9_GPIO_NUM;
393     config.pin_xclk = XCLK_GPIO_NUM;
394     config.pin_pclk = PCLK_GPIO_NUM;
395     config.pin_vsync = VSYNC_GPIO_NUM;
396     config.pin_href = HREF_GPIO_NUM;
397     config.pin_sscb_sda = SIOD_GPIO_NUM;
398     config.pin_sscb_scl = SIOC_GPIO_NUM;
399     config.pin_pwdn = PWDN_GPIO_NUM;
400     config.pin_reset = RESET_GPIO_NUM;
401     config.xclk_freq_hz = 20000000;
402     config.pixel_format = PIXFORMAT_JPEG;
403
404     config.frame_size = FRAMESIZE_VGA;
405     config.jpeg_quality = 10;
406     config.fb_count = 1;
407
408     // camera init
409     esp_err_t err = esp_camera_init(&config);
410     if (err != ESP_OK)
411     {
412         Serial.printf("Camera init failed with error 0x%x", err);
413         return;
414     }
415
416     if (psramFound())
417     {
418         heap_caps_malloc_extmem_enable(20000);
419         Serial.printf("PSRAM initialized. malloc to take memory from psram above this size");
420     }
421 }
422
423 void sendCameraPicture()
424 {
425     if (cameraClientId == 0)
426     {
427         return;
428     }
429     unsigned long startTime1 = millis();
430     //capture a frame
431     camera_fb_t * fb = esp_camera_fb_get();
432     if (!fb)
433     {
434         Serial.println("Frame buffer could not be acquired");
435         return;
436     }
437
438     unsigned long startTime2 = millis();
439     wsCamera.binary(cameraClientId, fb->buf, fb->len);
440     esp_camera_fb_return(fb);
441
442     //wait for message to be delivered
443     while (true)
444     {
445         AsyncWebSocketClient * clientPointer = wsCamera.client(cameraClientId);
446         if (!clientPointer || !(clientPointer->queueIsFull()))
447         {
448             break;
449         }
450         delay(1);
451     }
452
453     unsigned long startTime3 = millis();
454     Serial.printf("Time taken Total: %d|%d|%d\n", startTime3 - startTime1, startTime2 - startTime1, startTime3 - startTime2);
455 }
456
457 void setUpPinModes()
458 {
459     //Set up PWM
460     ledcSetup(PWMSpeedChannel, PWMFreq, PWMResolution);
461     ledcSetup(PWMLightChannel, PWMFreq, PWMResolution);

```

```

462
463   for (int i = 0; i < motorPins.size(); i++)
464   {
465       pinMode(motorPins[i].pinEn, OUTPUT);
466       pinMode(motorPins[i].pinIN1, OUTPUT);
467       pinMode(motorPins[i].pinIN2, OUTPUT);
468
469       /* Attach the PWM Channel to the motor enb Pin */
470       ledcAttachPin(motorPins[i].pinEn, PWMSpeedChannel);
471   }
472   moveCar(STOP);
473
474   pinMode(LIGHT_PIN, OUTPUT);
475   ledcAttachPin(LIGHT_PIN, PWMLightChannel);
476 }
477
478
479 void setup(void)
480 {
481     setUpPinModes();
482     Serial.begin(115200);
483
484     WiFi.mode(WIFI_STA);
485     WiFi.begin(ssid, password);
486     while (WiFi.status() != WL_CONNECTED) {
487         delay(1000);
488         Serial.println("Connecting to WiFi..");
489     }
490
491     //Serial.println(WiFi.localIP());
492     // Print ESP32 Local IP Address
493     //Serial.println(WiFi.localIP());
494
495     //WiFi.softAP(ssid, password);
496     //IPAddress IP = WiFi.softAPIP();
497     //Serial.print("AP IP address: ");
498     //Serial.println(IP);
499
500     server.on("/", HTTP_GET, handleRoot);
501     server.onNotFound(handleNotFound);
502
503     wsCamera.onEvent(onCameraWebSocketEvent);
504     server.addHandler(&wsCamera);
505
506     wsCarInput.onEvent(onCarInputWebSocketEvent);
507     server.addHandler(&wsCarInput);
508
509     server.begin();
510     Serial.println("HTTP server started");
511
512     setupCamera();
513 }
514
515
516 void loop()
517 {
518     wsCamera.cleanupClients();
519     wsCarInput.cleanupClients();
520     sendCameraPicture();
521     Serial.printf("SPIRam Total heap %d, SPIRam Free Heap %d\n", ESP.getPsramSize(), ESP.getFreePsram());
522 }

```