



ELETRÔNICA EMBARCADA COM MICROCONTROLADORES

ATIVIDADE 2 - ACIONANDO MOTORES ATRAVÉS DA INTERNET

PROFESSOR GUSTAVO FERREIRA PALMA

PÓS-GRADUANDA LIZ LENE DO PRADO PINTO

Porto Alegre, 03 de maio de 2024.

## ATIVIDADE 2 - ACIONANDO MOTORES ATRAVÉS DA INTERNET

Utilizando os conceitos abordados em aula, crie um firmware que seja capaz de:

- Via MQTT receber o sentido de giro do motor (horário/anti-horário);
- Via MQTT acionar o motor na intensidade desejada (controle PWM).

### IMPORTANTE!

- Os códigos devem ser desenvolvidos utilizando o SDK padrão C/C++ da Raspberry Pi Pico;
- Apenas o Link para o projeto no github deve ser incluído na tarefa;
- A entrega deve obedecer o prazo estipulado
- O projeto base disponibilizado na página do curso pode ser utilizado como base no desenvolvimento dessa atividade

Com base no exemplo da Aula 03, foi realizada a execução da Atividade 2, integrando o Software Visual Studio Code, o protocolo MQTT com o PuTTY (software de emulação de terminal).

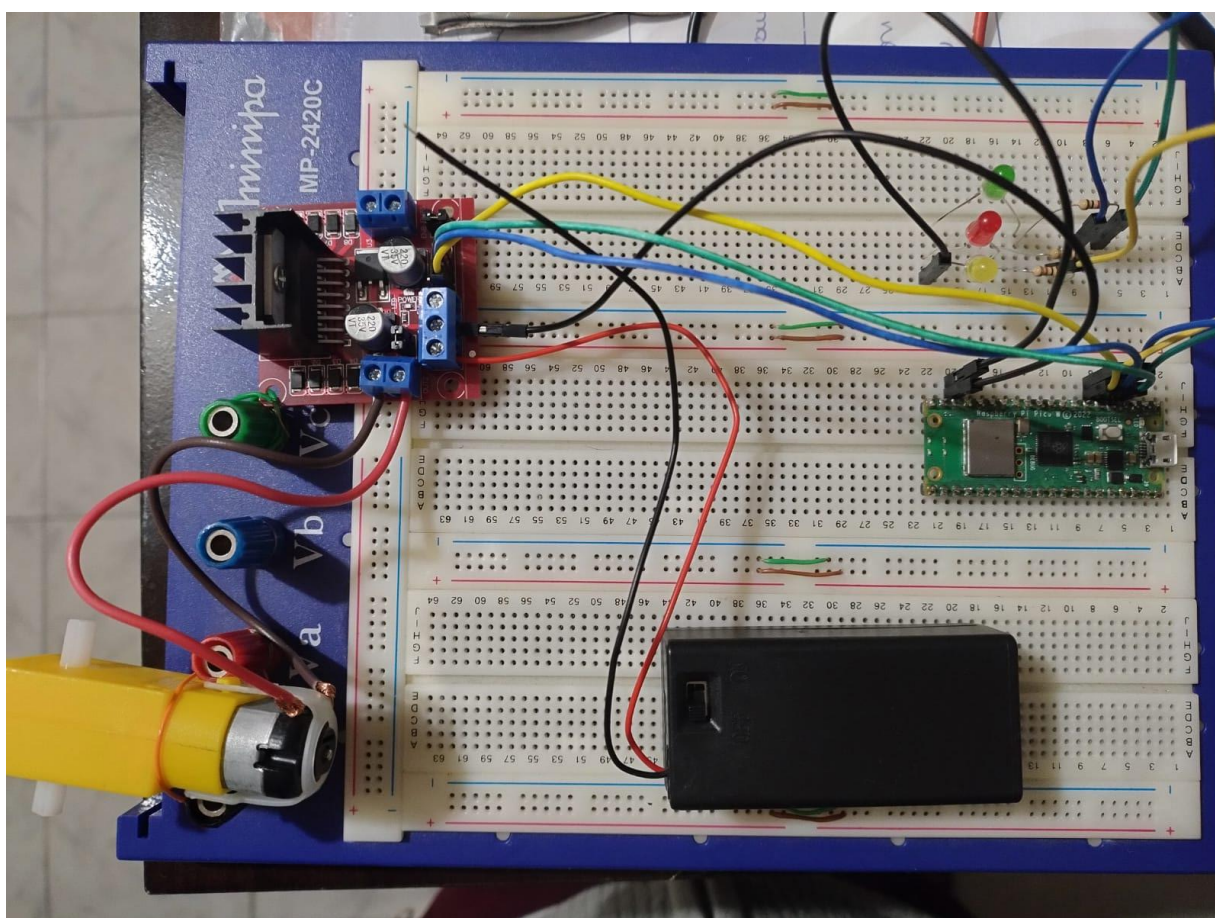


Figura 1 – Circuito para controlar o motor DC usando o Módulo de driver de motor L298N Através da placa Raspberry Pi Pico W

## 1) Descrição do código do arquivo main.c

```
1)  #include <stdio.h>
2)  #include "pico/stdlib.h"
3)  #include "pico/cyw43_arch.h"
4)  #include "hardware/gpio.h"
5)  #include "hardware/pwm.h"
6)  #include "bsp/board.h"
7)  #include "pico/multicore.h"
8)  #include "lwip/apps/mqtt.h"
9)
10) #define WIFI_SSID "RESOORTS 2.4G"
11) #define WIFI_PASSWORD "D3&M4@Yi5S"
12) #define MQTT_SERVER "54.146.113.169" //"broker.emqx.io"
13)
14) #define PIN_PWM 4 //18 fio amarelo do pwm = EN_A
15) #define PIN_MOTOR_A 2 //16 = A1
16) #define PIN_MOTOR_B 3 //17 = A2
17)
18) struct mqtt_connect_client_info_t mqtt_client_info=
19) {
20)     "<RA>/pico_w", /*client id*/
21)     NULL, /* user */
22)     NULL, /* pass */
23)     0, /* keep alive */
24)     NULL, /* will_topic */
25)     NULL, /* will_msg */
26)     0, /* will_qos */
27)     0 /* will_retain */
28) #if LWIP_ALTCP && LWIP_ALTCP_TLS
29)     , NULL
30) #endif
31) };
32)
33) static void mqtt_incoming_data_cb(void *arg, const u8_t *data, u16_t
len, u8_t flags) {
34)     printf("data: %s\n",data);
35) }
36)
37) static void mqtt_incoming_publish_cb(void *arg, const char *topic, u32_t
tot_len) {
38)     printf("topic %s\n", topic);
39) }
40)
41) static void mqtt_request_cb(void *arg, err_t err) {
42)     printf(("MQTT client request cb: err %d\n", (int)err));
43) }
44)
45) static void mqtt_connection_cb(mqtt_client_t *client, void *arg,
mqtt_connection_status_t status) {
```

```

46)     printf(("MQTT client connection cb: status %d\n", (int)status));
47) }
48)
49) // Variáveis globais para armazenar o sentido e intensidade do motor
50) volatile bool motor_clockwise = true; // Sentido horário como padrão
51) volatile uint16_t motor_intensity = 0; // Intensidade do motor (0 a
65535)
52)
53) // Mutex para sincronização de acesso às variáveis globais
54) mutex_t motor_mutex;
55)
56) // Função para controlar o motor baseado nos valores recebidos via MQTT
57) void motor_control_task() {
58)
59)     // Implemente a conexão MQTT e a subscrição ao tópico MQTT_TOPIC
aqui
60)
61)     while (true) {
62)
63)         uint slice_num = pwm_gpio_to_slice_num(PIN_PWM);
64)         uint16_t motor_intensity = 32768; // Intensidade do PWM, entre 0
e 65535
65)         // Receba a mensagem MQTT contendo o sentido e a intensidade do
motor
66)         // Atualize as variáveis globais motor_clockwise e
motor_intensity
67)         // Lembre-se de sincronizar o acesso às variáveis globais usando
o mutex
68)
69)         // Controle do sentido do motor
70)         mutex_enter_blocking(&motor_mutex);
71)         if (motor_clockwise) {
72)             gpio_put(PIN_MOTOR_A, 1);
73)             gpio_put(PIN_MOTOR_B, 0);
74)         } else {
75)             gpio_put(PIN_MOTOR_A, 0);
76)             gpio_put(PIN_MOTOR_B, 1);
77)         }
78)         mutex_exit(&motor_mutex);
79)
80)         // Controle da intensidade do motor via PWM
81)         pwm_set_chan_level(slice_num, pwm_gpio_to_channel(PIN_PWM),
motor_intensity);
82)
83)         sleep_ms(100); // Aguarde um curto período antes de verificar
novamente
84)     }
85) }
86)

```

```

87)  int main() {
88)
89)      stdio_init_all();
90)
91)      //para utilizar o EN_A como um I/O simples descomente a linha de
baixo
92)      //gpio_init(EN_A);
93)      gpio_set_function(PIN_PWM, GPIO_FUNC_PWM);
94)      uint slice_num = pwm_gpio_to_slice_num(PIN_PWM);
95)      uint16_t motor_intensity = 32768; // Intensidade do PWM, entre 0 e
65535
96)      pwm_config config = pwm_get_default_config();
97)      pwm_config_set_clkdiv(&config, 4.0f);
98)      pwm_set_wrap(slice_num, 65535);
99)      pwm_set_chan_level(slice_num, pwm_gpio_to_channel(PIN_PWM),
motor_intensity);
100)     pwm_set_enabled(slice_num, true);
101)
102)     // Inicialização dos pinos GPIO
103)     pwm_init(slice_num, &config, true);
104)     gpio_init(PIN_MOTOR_A);
105)     gpio_init(PIN_MOTOR_B);
106)     gpio_set_dir(PIN_MOTOR_A, GPIO_OUT);
107)     gpio_set_dir(PIN_MOTOR_B, GPIO_OUT);
108)
109)     gpio_put(PIN_MOTOR_A, true);
110)     gpio_put(PIN_MOTOR_B, false);
111)     pwm_set_gpio_level(PIN_PWM, 65535);
112)
113)     // Inicialização do mutex
114)     mutex_init(&motor_mutex);
115)
116)     bool subir = true;
117)     uint16_t duty_cycle = 0; //maximo 65535 - 100%
118)
119)     if (cyw43_arch_init())
120)     {
121)         printf("Falha ao iniciar chip wifi\n");
122)         return 1;
123)     }
124)
125)     cyw43_arch_enable_sta_mode();
126)     printf("Conectando ao %s\n", WIFI_SSID);
127)
128)     if (cyw43_arch_wifi_connect_timeout_ms(WIFI_SSID, WIFI_PASSWORD,
CYW43_AUTH_WPA2_AES_PSK, 30000))
129)     {
130)         printf("Falha ao conectar\n");
131)         return 1;

```

```

132)     }
133)     printf("Conectado ao Wifi %s\n", WIFI_SSID);
134)
135)     ip_addr_t addr;
136)     if(!ip4addr_aton(MQTT_SERVER, &addr)){
137)         printf("ip error\n");
138)         return 1;
139)     }
140)
141)     printf("Conectando ao MQTT\n");
142)
143)     mqtt_client_t* cliente_mqtt = mqtt_client_new();
144)
145)     mqtt_set_inpub_callback(cliente_mqtt, &mqtt_incoming_publish_cb,
&mqtt_incoming_data_cb, NULL);
146)
147)     err_t erro = mqtt_client_connect(cliente_mqtt, &addr, 1883,
&mqtt_connection_cb, NULL, &mqtt_client_info);
148)
149)     if (erro != ERR_OK)
150)     {
151)         printf("Erro de conexão\n");
152)         return 1;
153)     }
154)
155)     printf("Conectado ao MQTT!\n");
156)
157)     //para utilizar o EN_A como um I/O simples descomente a linha de
baixo
158)     //gpio_set_dir(EN_A, GPIO_OUT);
159)
160)     // Inicialização da tarefa de controle do motor no core secundário
161)     multicore_launch_core1(motor_control_task);
162)
163)     while(1) {
164)         if (subir){
165)             if (board_button_read()){
166)                 duty_cycle += 1000;
167)             }
168)             if (duty_cycle > 65535) {
169)                 subir = false;
170)             }
171)         }else {
172)             if (board_button_read()){
173)                 duty_cycle -= 1000;
174)             }
175)             if (duty_cycle < 100){
176)                 subir = true;
177)             }

```

```

178)         }
179)         pwm_set_gpio_level(PIN_PWM, duty_cycle);
180)
181)         sleep_ms(200);
182)     }
183)     //para utilizar o EN_A como um I/O simples descomente a linha de
baixo
184)     /*
185)     while(1) {
186)         if (board_button_read) {
187)             gpio_put(EN_A, false);
188)             sleep_ms(20);
189)             gpio_put(A1, false);
190)             gpio_put(A2, true);
191)             gpio_put(EN_A, true);
192)         } else {
193)             gpio_put(EN_A, false);
194)             sleep_ms(20);
195)             gpio_put(A1, true);
196)             gpio_put(A2, false);
197)             gpio_put(EN_A, true);
198)         }
199)         sleep_ms(200);
200)     }
201)     */
202) }

```

## 2) Descrição do código do arquivo CMakeLists.txt

```

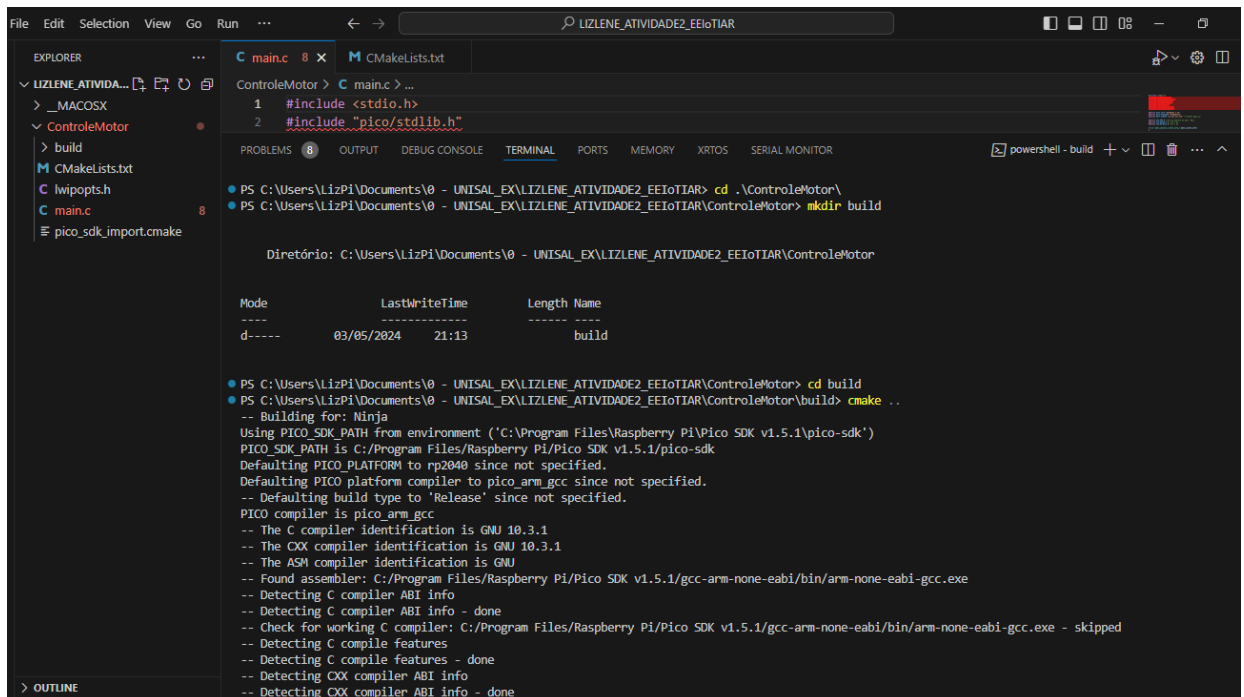
1)  cmake_minimum_required(VERSION 3.12)
2)
3)  include(pico_sdk_import.cmake)
4)
5)  project(motor C CXX ASM)
6)
7)  set(CMAKE_C_STANDARD 11)
8)  set(CMAKE_CXX_STANDARD 17)
9)
10) #Caso seja a Raspberry Pi Pico W
11) set(PICO_BOARD pico_w)
12)
13) pico_sdk_init()
14) include(pico_sdk_import.cmake)
15)
16) add_executable(motor main.c)
17)
18) pico_enable_stdio_usb(motor 1)
19) pico_enable_stdio_uart(motor 0)

```

```
20)
21) target_include_directories(motor PRIVATE
22)     ${CMAKE_CURRENT_LIST_DIR}
23) )
24)
25) target_include_directories(motor PUBLIC
26)     ${CMAKE_CURRENT_LIST_DIR}/pico-sdk/src
27) )
28)
29) target_link_libraries(motor
30)     tinyusb_board
31)     hardware_pwm
32)     pico_cyw43_arch_lwip_threadsafe_background
33)     pico_stdlib
34)     pico_lwip_mqtt
35)     pico_multicore
36) )
37) pico_add_extra_outputs(motor)
38)
```

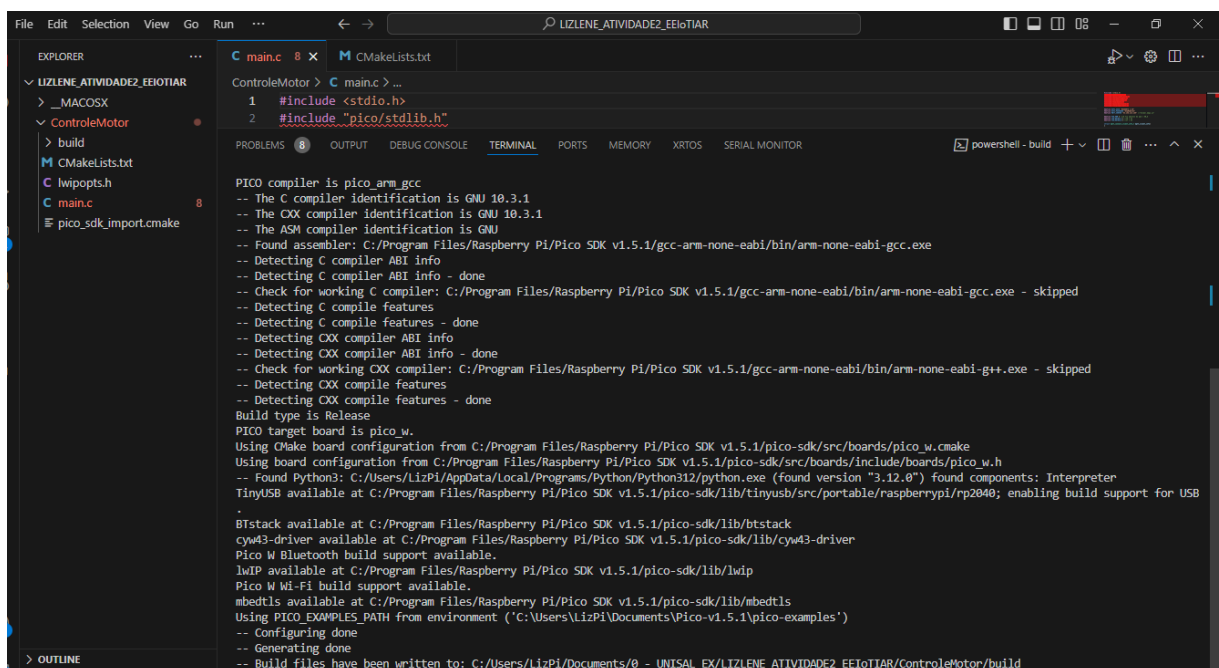


### 3) Preparação da estrutura e compilação do código



```
File Edit Selection View Go Run ...  
LIZLENE_ATIVIDADE2_EEIoTAR  
EXPLORER  
LIZLENE_ATIVIDADE2_EEIoTAR  
  _MACOSX  
  ControleMotor  
    build  
    CMakeLists.txt  
    lwipopts.h  
    main.c  
    pico_sdk_import.cmake  
  OUTLINE  
C main.c 8 x M CMakeLists.txt  
ControleMotor > C main.c > ...  
1 #include <stdio.h>  
2 #include "pico/stdlib.h"  
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR  
PS C:\Users\LizPi\Documents\0 - UNISAL_EX\LIZLENE_ATIVIDADE2_EEIoTAR\ControleMotor> cd .\ControleMotor\  
PS C:\Users\LizPi\Documents\0 - UNISAL_EX\LIZLENE_ATIVIDADE2_EEIoTAR\ControleMotor> mkdir build  
Diretório: C:\Users\LizPi\Documents\0 - UNISAL_EX\LIZLENE_ATIVIDADE2_EEIoTAR\ControleMotor  
Mode LastWriteTime Length Name  
d---- 03/05/2024 21:13 build  
PS C:\Users\LizPi\Documents\0 - UNISAL_EX\LIZLENE_ATIVIDADE2_EEIoTAR\ControleMotor> cd build  
PS C:\Users\LizPi\Documents\0 - UNISAL_EX\LIZLENE_ATIVIDADE2_EEIoTAR\ControleMotor> cmake ..  
-- Building for: Ninja  
Using PICO_SDK_PATH from environment ('C:\Program Files\Raspberry Pi\Pico SDK v1.5.1\pico-sdk')  
PICO_SDK_PATH is C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk  
Defaulting PICO_PLATFORM to rp2040 since not specified.  
Defaulting PICO platform compiler to pico_arm_gcc since not specified.  
-- Defaulting build type to 'Release' since not specified.  
PICO compiler is pico_arm_gcc  
-- The C compiler identification is GNU 10.3.1  
-- The CXX compiler identification is GNU 10.3.1  
-- The ASM compiler identification is GNU  
-- Found assembler: C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/gcc-arm-none-eabi/bin/arm-none-eabi-gcc.exe  
-- Detecting C compiler ABI info  
-- Detecting C compiler ABI info - done  
-- Check for working C compiler: C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/gcc-arm-none-eabi/bin/arm-none-eabi-gcc.exe - skipped  
-- Detecting C compile features  
-- Detecting C compile features - done  
-- Detecting CXX compiler ABI info  
-- Detecting CXX compiler ABI info - done
```

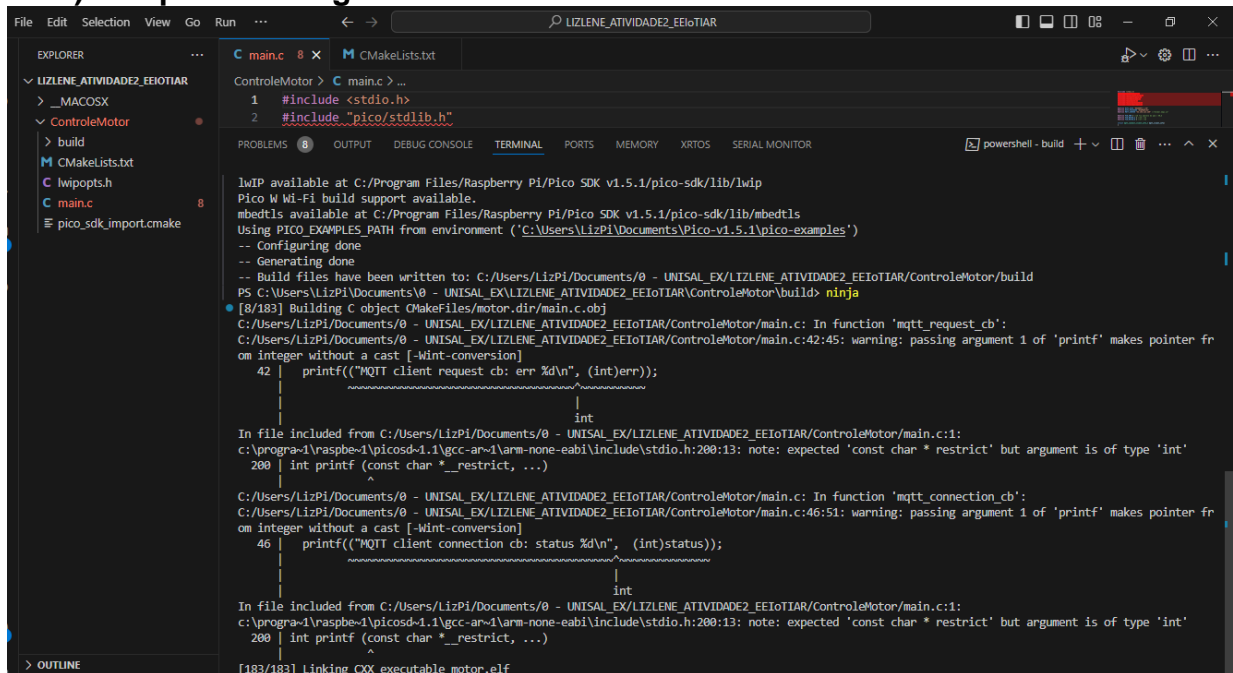
Figura 2 – Compilação



```
File Edit Selection View Go Run ...  
LIZLENE_ATIVIDADE2_EEIoTAR  
EXPLORER  
LIZLENE_ATIVIDADE2_EEIoTAR  
  _MACOSX  
  ControleMotor  
    build  
    CMakeLists.txt  
    lwipopts.h  
    main.c  
    pico_sdk_import.cmake  
  OUTLINE  
C main.c 8 x M CMakeLists.txt  
ControleMotor > C main.c > ...  
1 #include <stdio.h>  
2 #include "pico/stdlib.h"  
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR  
PICO compiler is pico_arm_gcc  
-- The C compiler identification is GNU 10.3.1  
-- The CXX compiler identification is GNU 10.3.1  
-- The ASM compiler identification is GNU  
-- Found assembler: C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/gcc-arm-none-eabi/bin/arm-none-eabi-gcc.exe  
-- Detecting C compiler ABI info  
-- Detecting C compiler ABI info - done  
-- Check for working C compiler: C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/gcc-arm-none-eabi/bin/arm-none-eabi-gcc.exe - skipped  
-- Detecting C compile features  
-- Detecting C compile features - done  
-- Detecting CXX compiler ABI info  
-- Detecting CXX compiler ABI info - done  
-- Check for working CXX compiler: C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/gcc-arm-none-eabi/bin/arm-none-eabi-g++.exe - skipped  
-- Detecting CXX compile features  
-- Detecting CXX compile features - done  
Build type is Release  
PICO target board is pico_w.  
Using CMake board configuration from C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/src/boards/pico_w.cmake  
Using board configuration from C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/src/boards/include/boards/pico_w.h  
-- Found Python3: C:/Users/LizPi/AppData/Local/Programs/Python/Python312/python.exe (found version "3.12.0") found components: Interpreter  
TinyUSB available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040; enabling build support for USB  
BTstack available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/btstack  
cyw43-driver available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/cyw43-driver  
Pico W Bluetooth build support available.  
lwIP available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/lwip  
Pico W Wi-Fi build support available.  
mbedtls available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/mbedtls  
Using PICO_EXAMPLES_PATH from environment ('C:/Users/LizPi/Documents/Pico-v1.5.1/pico-examples')  
-- Configuring done  
-- Generating done  
-- Build files have been written to: C:/Users/LizPi/Documents/0 - UNISAL_EX\LIZLENE_ATIVIDADE2_EEIoTAR\ControleMotor\build
```

Figura 3 – Continuação da compilação

#### 4) Output do código



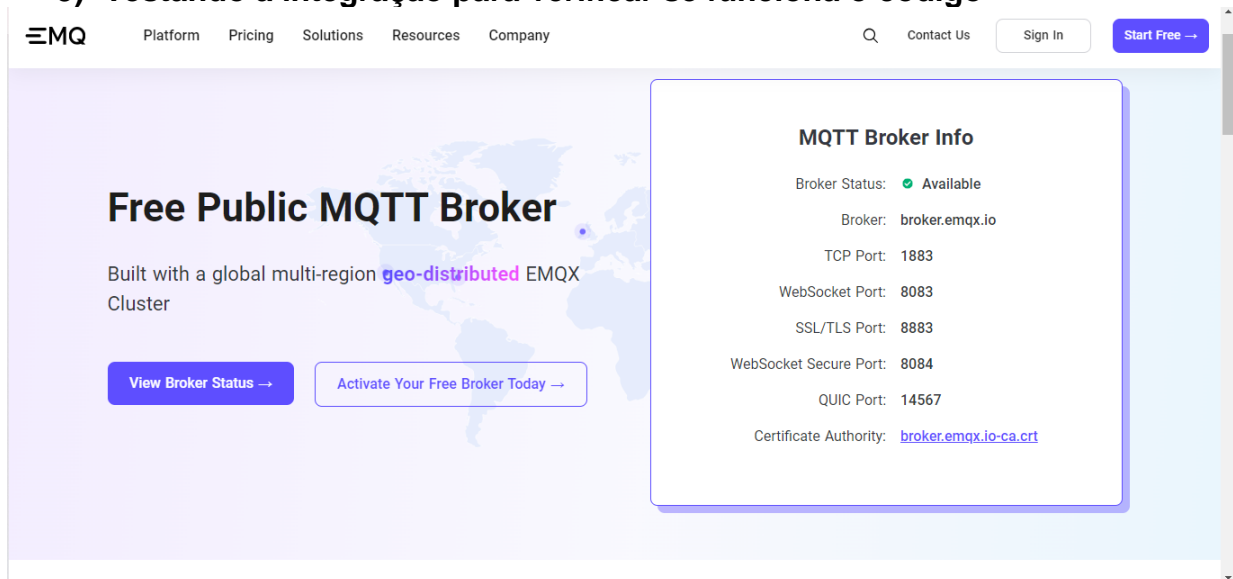
```
ControlMotor > C main.c > ...
1 #include <stdio.h>
2 #include "pico/stdlib.h"

PROBLEMS 0 OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY XRTOS SERIAL MONITOR
powershell - build + v ...

lwIP available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/lwip
Pico Wi-Fi build support available.
mbedtls available at C:/Program Files/Raspberry Pi/Pico SDK v1.5.1/pico-sdk/lib/mbedtls
Using PICO_EXAMPLES_PATH from environment ('C:/Users/LizPi/Documents/Pico-v1.5.1/pico-examples')
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/LizPi/Documents/0 - UNISAL_EX/LIZLENE_ATIVIDADE2_EEIOTIAR/ControlMotor/build
PS C:/Users/LizPi/Documents/0 - UNISAL_EX/LIZLENE_ATIVIDADE2_EEIOTIAR/ControlMotor/build> ninja
[8/183] Building C object CMakeFiles/motor.dir/main.c.obj
C:/Users/LizPi/Documents/0 - UNISAL_EX/LIZLENE_ATIVIDADE2_EEIOTIAR/ControlMotor/main.c: In function 'mqtt_request_cb':
C:/Users/LizPi/Documents/0 - UNISAL_EX/LIZLENE_ATIVIDADE2_EEIOTIAR/ControlMotor/main.c:42:45: warning: passing argument 1 of 'printf' makes pointer fr
om integer without a cast [-Wint-conversion]
42 | printf("MQTT client request cb: err %d\n", (int)err));
   |                                     ^
   |                                     |
   |                                     int
In file included from C:/Users/LizPi/Documents/0 - UNISAL_EX/LIZLENE_ATIVIDADE2_EEIOTIAR/ControlMotor/main.c:1:
c:/progra-1/raspbe-1/picosd-1.1/gcc-arm-none-eabi/include/stdio.h:200:13: note: expected 'const char * restrict' but argument is of type 'int'
200 | int printf(const char *__restrict, ...)
    |             ^
C:/Users/LizPi/Documents/0 - UNISAL_EX/LIZLENE_ATIVIDADE2_EEIOTIAR/ControlMotor/main.c: In function 'mqtt_connection_cb':
C:/Users/LizPi/Documents/0 - UNISAL_EX/LIZLENE_ATIVIDADE2_EEIOTIAR/ControlMotor/main.c:46:51: warning: passing argument 1 of 'printf' makes pointer fr
om integer without a cast [-Wint-conversion]
46 | printf("MQTT client connection cb: status %d\n", (int)status));
   |                                     ^
   |                                     |
   |                                     int
In file included from C:/Users/LizPi/Documents/0 - UNISAL_EX/LIZLENE_ATIVIDADE2_EEIOTIAR/ControlMotor/main.c:1:
c:/progra-1/raspbe-1/picosd-1.1/gcc-arm-none-eabi/include/stdio.h:200:13: note: expected 'const char * restrict' but argument is of type 'int'
200 | int printf(const char *__restrict, ...)
    |             ^
[183/183] Linking CXX executable motor.elf
```

Figura 4 – Utilizando o comando “ninja”, para gerar o arquivo .uf2

#### 5) Testando a integração para verificar se funciona o código



**Free Public MQTT Broker**

Built with a global multi-region **geo-distributed** EMQX Cluster

[View Broker Status →](#) [Activate Your Free Broker Today →](#)

**MQTT Broker Info**

Broker Status: ● Available

Broker: broker.emqx.io

TCP Port: 1883

WebSocket Port: 8083

SSL/TLS Port: 8883

WebSocket Secure Port: 8084

QUIC Port: 14567

Certificate Authority: [broker.emqx.io-ca.crt](https://broker.emqx.io-ca.crt)

Figura 5 – MQTT Broker Info com os dados necessários para a comunicação na internet



```
COM6 - PuTTY

Conectando ao RESOORTS 2.4G
Conectado ao Wifi RESOORTS 2.4G
Conectando ao MQTT
Conectado ao MQTT!
```

Figura 6 – Verificando a conexão através do Putty com o circuito

## 6) Simulação do giro do motor

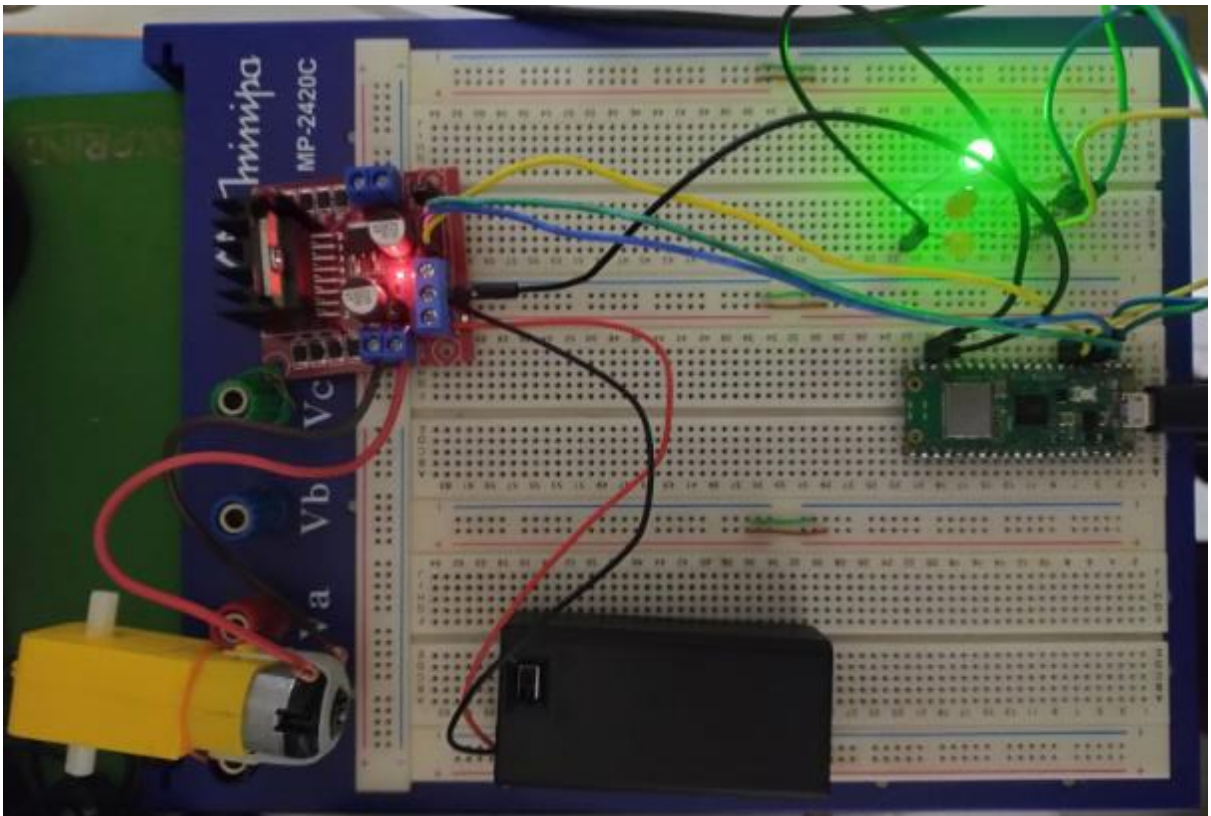


Figura 7 – Giro para a direita

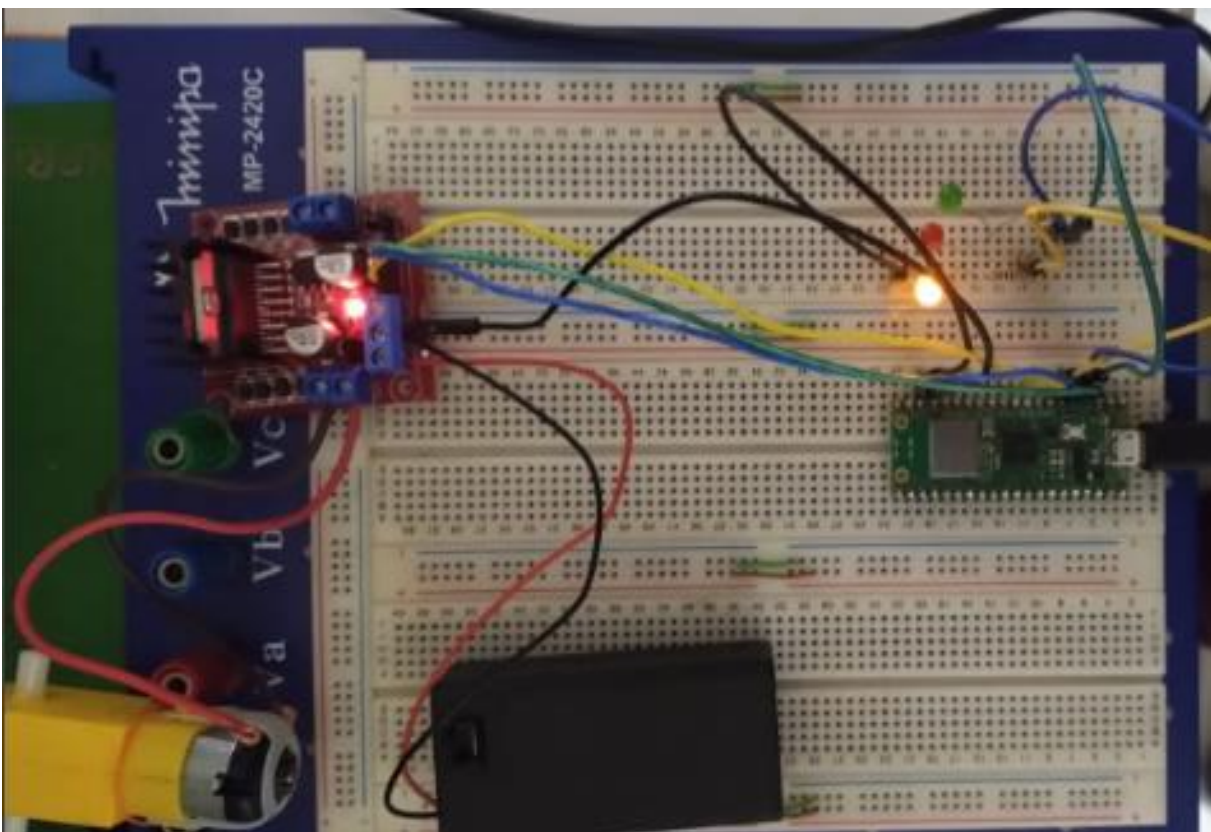


Figura 8 – Giro para a esquerda

## **7) Referência bibliográfica**

**7.1)** [https://ava.ead.unisal.br/pluginfile.php/60694/mod\\_resource/content/1/Aula-2.pdf](https://ava.ead.unisal.br/pluginfile.php/60694/mod_resource/content/1/Aula-2.pdf)

**7.2)** [https://ava.ead.unisal.br/pluginfile.php/60694/mod\\_resource/content/1/Aula-3.pdf](https://ava.ead.unisal.br/pluginfile.php/60694/mod_resource/content/1/Aula-3.pdf)

**7.3)** <https://www.youtube.com/watch?v=ZM6bmV93UvY>

**7.4)** <https://www.youtube.com/watch?v=xVuVQrlpuxQ>

**7.5)** [https://microdigisoft.com/control-dc-motor-using-l298n-raspberry-pi-pico-micropython/#google\\_vignette](https://microdigisoft.com/control-dc-motor-using-l298n-raspberry-pi-pico-micropython/#google_vignette)

**7.6)** <https://capsistema.com.br/index.php/2022/06/03/control-o-motor-dc-usando-o-driver-l298n-com-raspberry-pi-pico-e-micropython/>