UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Guilherme Lettmann Penha - 23100372;

Relatório Prova 1: Implementação da Quesãto 2 no ESP

FLorianópolis
2025

1 INTRODUÇÃO

Este relatório apresenta o desenvolvimento do código referente à Questão 2 da Prova 1, focado no controle de atuadores. O objetivo principal foi adaptar e corrigir a implementação existente, garantindo que o software pudesse ser executado de forma confiável em um sistema embarcado (SoC).

Durante o processo, foram realizados ajustes na lógica de controle e na estrutura do código, visando compatibilidade com a plataforma embarcada, otimização do fluxo de execução e registro eficiente de informações de estado por meio de logs. O relatório detalha as modificações realizadas, a abordagem utilizada para integração com o hardware e os resultados obtidos após a adaptação para o ambiente embarcado.

2 PROCEDIMENTOS

2.1 CONSERTOS DO CÓDIGO DA PROVA

Existiam 3 erros principais nos códigos que foram realizados durante a P1. No arquivo "*LED.cpp*", código fonte da classe LED, que herda características da classe *Actuators*, existia um 0 que foi inserido por *missclick*, que foi removido.

O outro erro foi que a função *main* foi declarada 2 vezes por engano no arquivo "*main.cpp*", que também foi consertado.

Eu também havia esquecido de escrever a função *getSpeed* da classe *Motors*, função simples que implementei mentalmente para "pegar"o valor da velocidade dos motores de forma simples e havia esquecido de escrever, a função escrita em "*Motors.cpp*"(com seu cabeçalho em "*Motors.h*") está referenciada no Anexo A.1.

2.2 ALTERAÇÕES PARA RODAR NO ESP

As maiores atualizações no código (tirando as alterações no "*CMakeList.txt*", onde adicionei todos os *Sources* do projeto, e a declaração "*extern c*"no "*main.c*"(que virou "*main.cpp*")) foram na comunicação entre programa e tela (cout para LOGI) e na estrutura de *loop* infinito.

Todas as funções que *printavam* algo tiveram sua estrutura alterada conforme o Anexo A.2 (além dos includes necessários). A sintaxe com cout funciona em alguns casos, mas não é o ideal para sistemas embarcados.

Além disso, o ESP-IDF não lida bem com as exceções, então elas foram comentadas para o funcionamento do código

3 RESULTADOS E DISCUSSÃO

Chega-se a conclusão que não é difícil realizar a portabilidade de um código em C++ para dentro de sistemas embarcados, a maior dificuldade é ter que criar a portabilidade, uma vez que o ambiente ESP-IDF não é preparado para isso, talvez seja mais fácil em algum outro ambiente (como o STM Cube IDE), que aceite fácilmente a portabilidade para C++.

Mesmo tendo alterado manualmente a estrutura do main e o arquivo Cmake, podemos verificar o funcionamento conforme os Anexo A.3 A.4

ANEXO A – ALTERAÇÕES DO CÓDIGO

```
int Motor::getSpeed() {
    return speed;
}
```

Figura A.1 – Função *getSpeed*.

```
void Motor::printStatus() {
    ESP_LOGI("MOTOR", "Tipo: Motor | ID: %d | Velocidade: %d", id, speed);
}
```

Figura A.2 – Sintaxe para *Serial Print*.

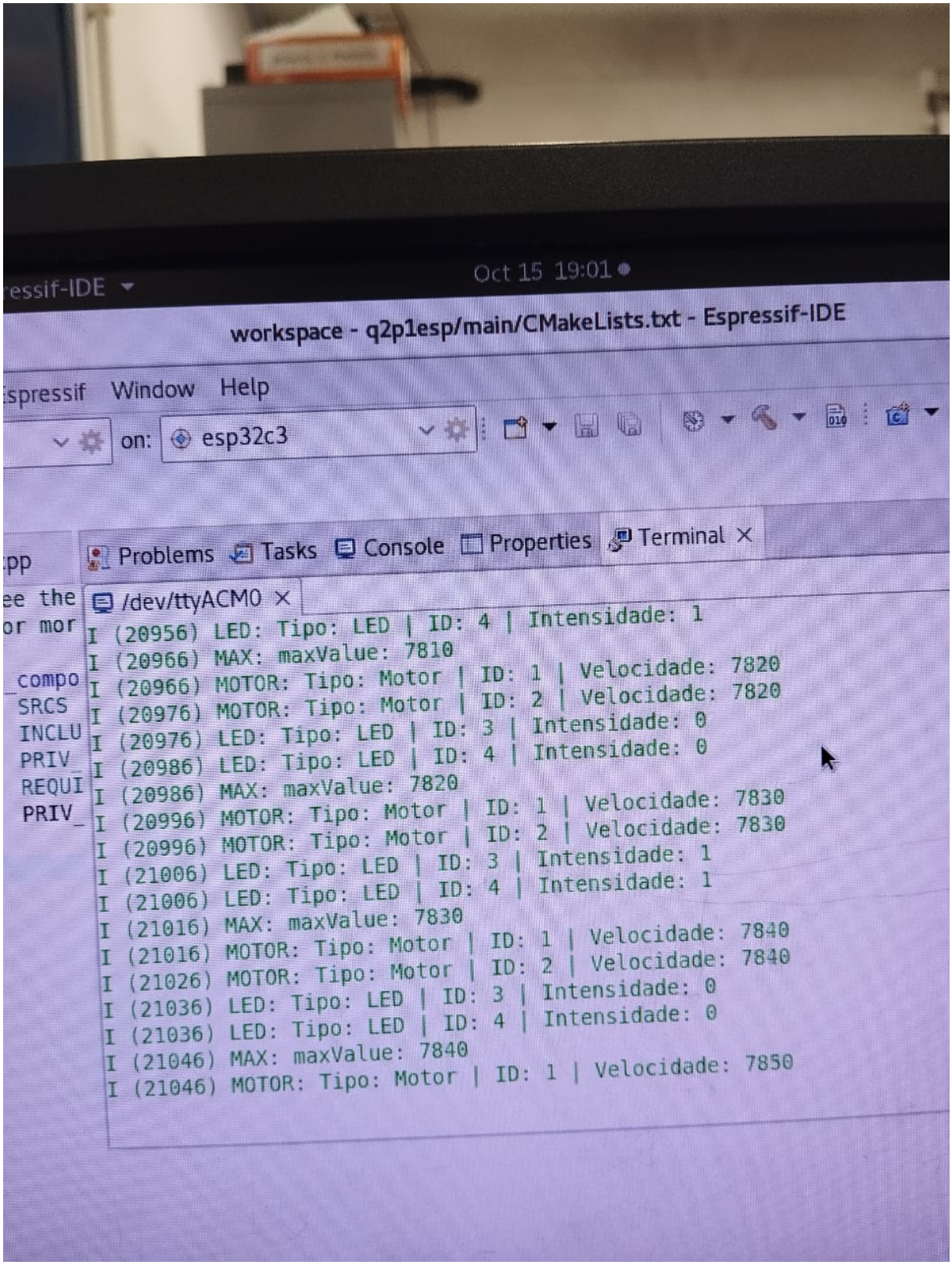

Figura A.3 – Terminal Funcionando.

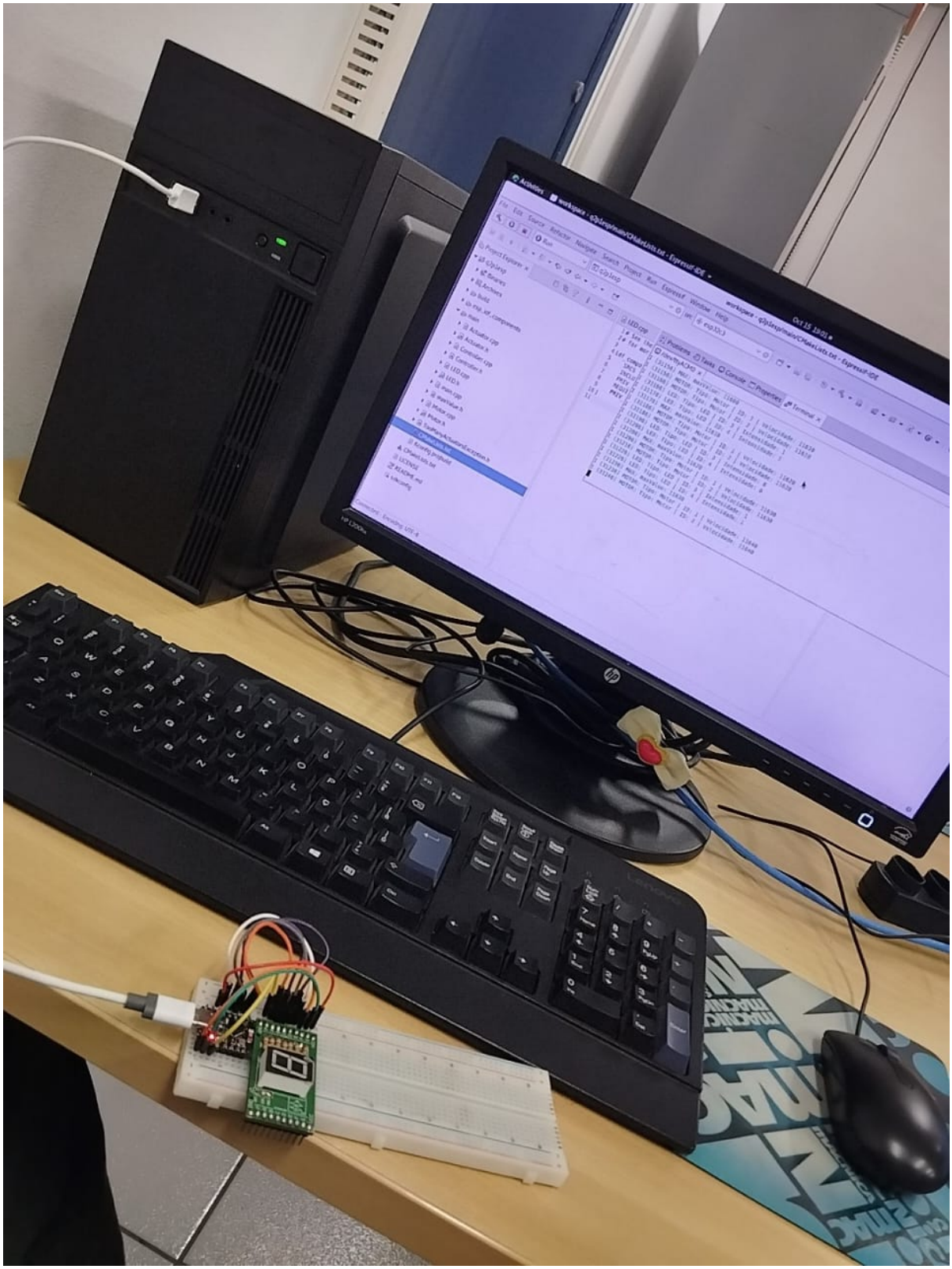

Figura A.4 – Circuito Montado.