Relatório de Atividade

Carrinho de Controle Remoto

AFONSO H. G. DE SOUZA, ALICE H. T. SILVA, GIOVANNA M. P. ZAGO, LORRAN P. V. DUTRA, MICHAEL C. OLIVEIRA, SARAH C. DE OLIVEIRA

Petianos responsáveis pelo projeto, em ordem alfabética

Resumo

O projeto do Carrinho de Cotrole Remoto foi elaborado no intuito de confeccionar um protótipo robusto baseado em outro ja existente no grupo PETEE. A ideia é controlar um carrinho remotamente através de um controle reutilizado com microcontrolador (Arduino), transmissor e receptor sem fio (NRF).

I. INTRODUÇÃO

O Carrinho de Controle Remoto foi feito como Projeto Calouro dos seis novos petianos de 2018/1. A grande motivação para este projeto foi o grande sucesso e utilidade do Projeto Calouro anterior de Carrinho de Controle nas atividades de extensão que possuia grandes problemas elétricos e mecânicos, ou seja, de robustez.

II. MATERIAIS E MÉTODOS

Materiais para construção do Controle:

- Controle de videogame com os analógicos funcionando
- Módulo NRF24L01+
- Regulador de tensão 3.3V AMS1117 SMD
- Fios finos coloridos
- Arduino ProMini 5V 16MHz (ou Arduino Nano que é um pouco maior)
- Bateria de Lipo 3,7V (uma célula) até 300mAh
- Capacitor Eletrolítico 100uF - 10V

Materiais para construção do Carrinho:

- Módulo NRF24L01+
- CI ATMEGA328P (CI do Arduino UNO)
- Resistor 10k
- Oscilador de cristal de 16MHz
- 2x capacitor cerâmico 22pF
- 2x capacitor cerâmico 100nF
- Capacitor Eletrolítico 1000uF - 6,3V
- Botão pequeno para reset
- LED Verde 3mm
- Resistor 400Ω
- Regulador de tensão para 5V
- Capacitor Eletrolítico 100uF - 10V
- Regulador de tensão 3.3V AMS1117 SMD
- Indutor de núcleo de ferrite
- Bateria de LIPO 7,4V (duas células) mínimo 1000mAh
- Placa de acrílico
- 2x motores com caixa de transmissão de 6-8V
- roda boba
- 2x rodas com pneu
- Borne de duas entradas azul (VCC)
- Borne de duas entradas preto (GND)
- Barramento de pinos tipo fêmea
- fios coloridos
- jumpers
- placa de fenolite
- chave de duas posições

II MATERIAIS E MÉTODOS

- Ponte H L298N

Foi utilizado um controle de playstation 2 estragado (Figura 2) mas com os analógicos em perfeito funcionamento, porque é utilizado apenas o analógico do controle. O controle foi montado como mostra a Figura 1.

É de extrema importância conferir as conecções com as definições de entrada e saída do código fonte.

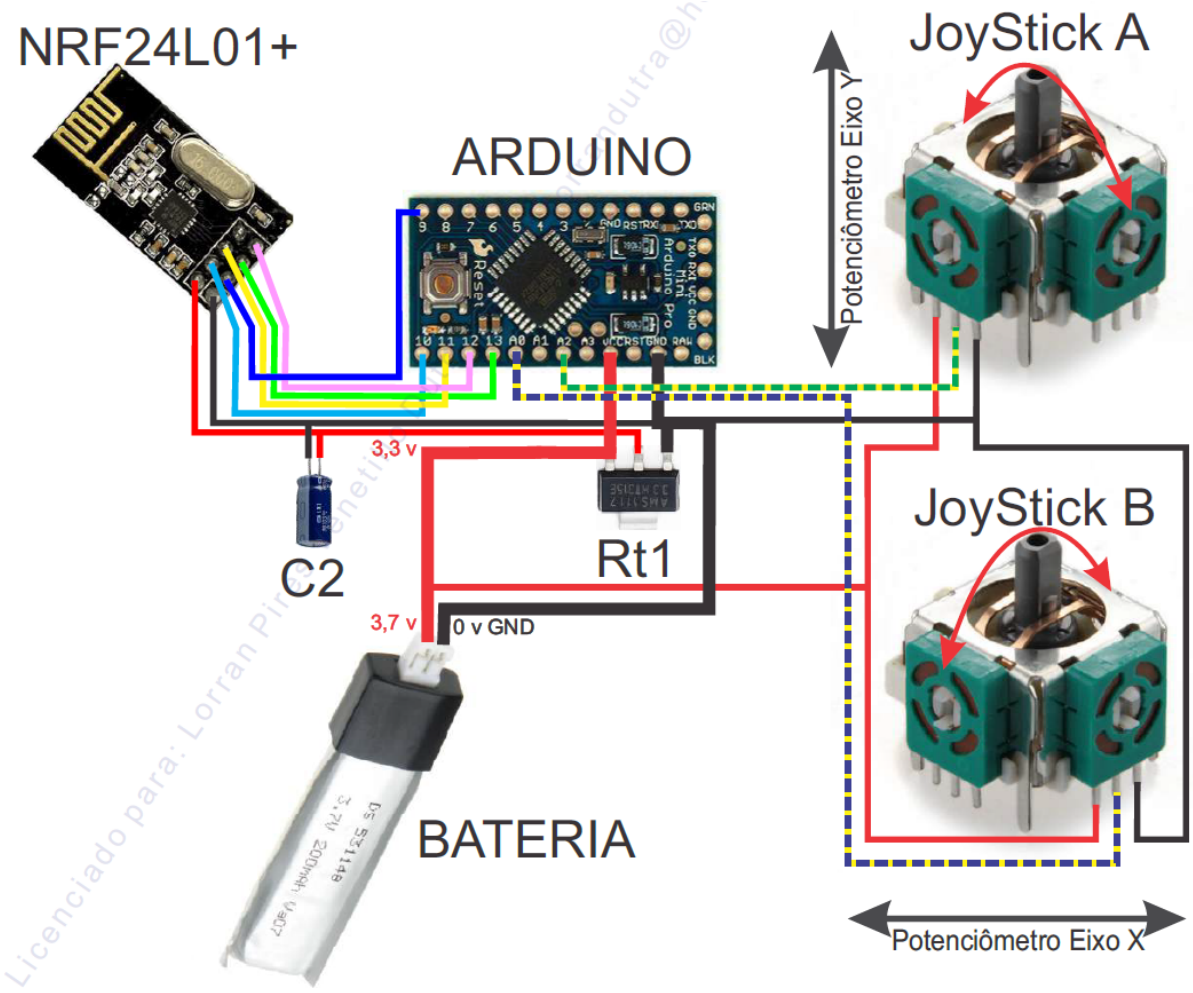

Figura 1: *Diagrama de Montagem do Controle*

Figura 2: *Controle Montado*

Para a confecção da carcaça do carrinho, inicialmente foi feito um protótipo de papelão e ao ser aprovado pela equipe, passou-se esse projeto para o programa CorelDraw. Com isso, o projeto foi enviado para uma empresa especializada em corte de acrilico. Com as peças cortadas, utilizou-se cola do tipo tecbond para montar a carcaça do carrinho.
A parte elétrica (placa de circuito impresso) foi elaborada no programa Eagle (Figura 3) e posteriormente importado na conta do PETEE do programa EasyEDA. A placa de circuito impresso ficou como mostra a Figura 4.

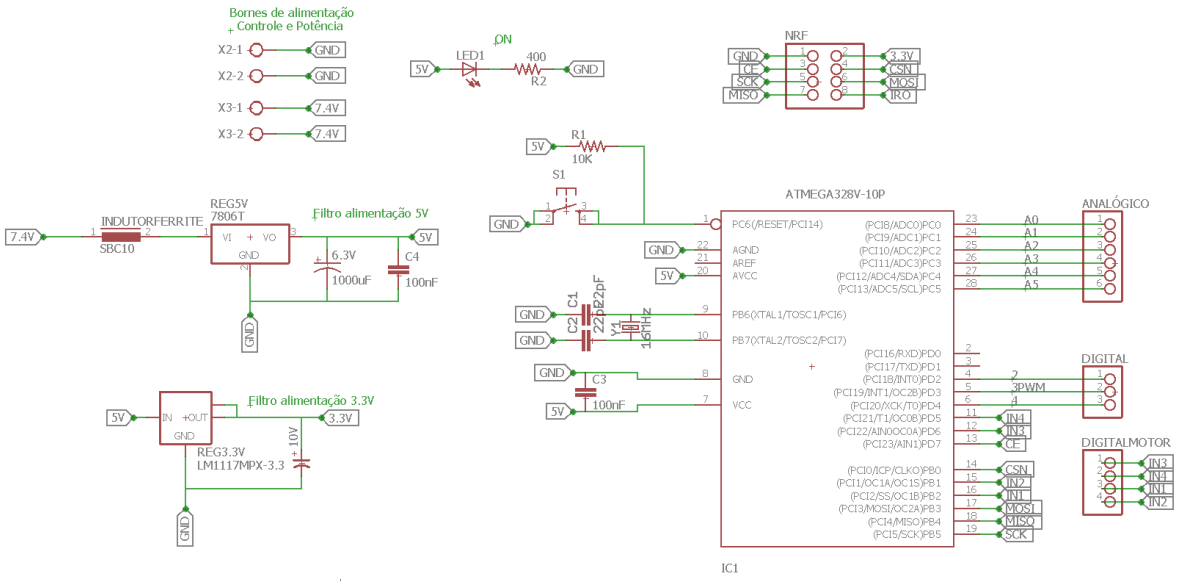

Figura 3: *Esquemático da Placa de Circuito Impresso*

II MATERIAIS E MÉTODOS

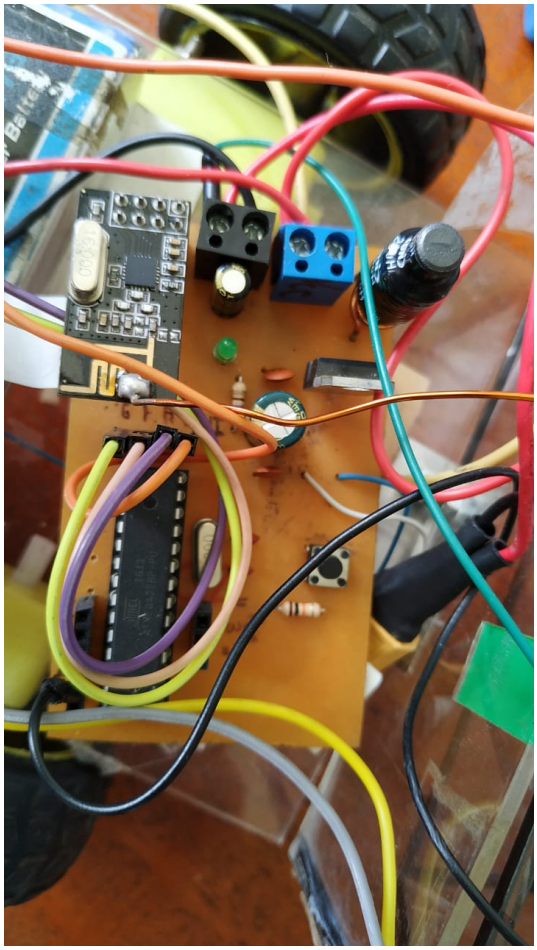

Figura 4: *Placa Pronta*

O interior do Carrinho de Controle Remoto ficou como mostra a Figura 5.

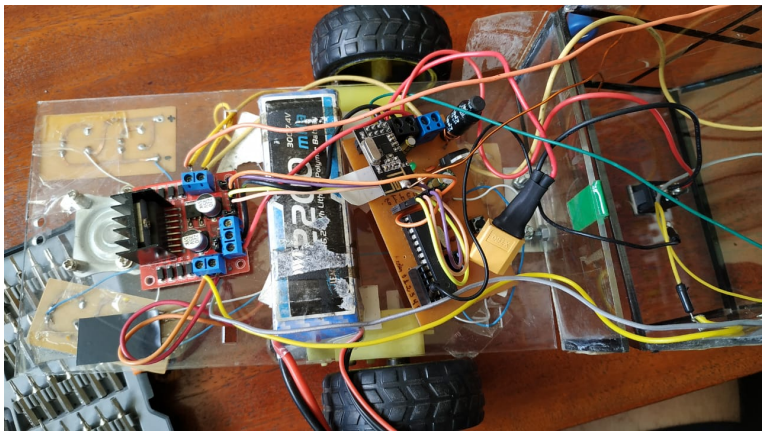

Figura 5: *Controle Montado*

Código Controle

```
1
```

```
2
3  #include <RF24.h>
4
5  //************** Definição dos Pinos **********************
6  #define pinCE    7
7  #define pinCSN   8
8  #define pinPot1  A1  //Frente e Reverso
9  #define pinPot2  A2  //Direito e Esquerdo
10 #define R3 4//botao r3 digital, tem que ser analogico?
11
12 struct tipoDadosRF
13 {
14    int pot1 = 512;
15    int pot2 = 512;
16    int pot3 = HIGH ;//boll ou int?
17 };
18 tipoDadosRF dadosRF;
19
20 RF24 radio(pinCE,pinCSN);
21
22 const uint64_t pipeOut = 0xE8E8F0F0E1LL;
23
24
25 void setup() {
26   //************** Controle do RF **********************
27   radio.begin();
28   radio.setPALevel( RF24_PA_LOW );      //RF24_PA_MIN   / RF24_PA_LOW /
      RF24_PA_HIGH / RF24_PA_MAX
29   radio.setDataRate( RF24_250KBPS );    //RF24_250KBPS / RF24_1MBPS   / RF24_2MBPS
30   radio.openWritingPipe(pipeOut);
31   Serial.begin(9600);
32   //Serial.print("dnisni");
33   pinMode(R3, INPUT_PULLUP);
34
35 }
36
37
38 void loop() {
39
40   //************** Controle do RF **********************
41   radio.write( &dadosRF, sizeof(tipoDadosRF) );
42
43   //************** Controle do Joystick ************
44   dadosRF.pot1 = analogRead(pinPot1);
45   dadosRF.pot2 = analogRead(pinPot2);
46   dadosRF.pot3 = digitalRead(R3);//e assim que pega o sinal?
47   //Serial.println(dadosRF.pot3);
48
49   //Serial.println(dadosRF.pot2);
50
51 }
```

II MATERIAIS E MÉTODOS

Código Carrinho

```
#include <RF24.h>

//*************** Definição dos Pinos **********************
#define pinCE  7
#define pinCSN 8
#define pinIN1 10
#define pinIN2 9
#define pinIN3 6
#define pinIN4 5
#define pinIN5 2 //pino que acendo o LED
//#define pinIN6   //"BOTAO R3/L3 CONTROLE"
#define pinIN7 4

struct tipoDadosRF
{
    int pot1 = 512;
    int pot2 = 512;
    int pot3 = HIGH;//boll ou int?
};
tipoDadosRF dadosRF;

RF24 radio(pinCE,pinCSN);

const uint64_t pipeOut = 0xE8E8F0F0E1LL;


//*************** Controle do Carro ************
int pDireita  = 100;
int pEsquerda = 100;


void setup() {
  //*************** Controle do RF **********************
  radio.begin();
  radio.setPALevel( RF24_PA_LOW );      //RF24_PA_MIN   / RF24_PA_LOW /
   RF24_PA_HIGH / RF24_PA_MAX
  radio.setDataRate( RF24_250KBPS );    //RF24_250KBPS / RF24_1MBPS   / RF24_2MBPS
  radio.openReadingPipe(1, pipeOut);
  radio.startListening();
  Serial.begin(9600);


  //*************** Controle do Carro ************
  pinMode(pinIN1, OUTPUT);
  pinMode(pinIN2, OUTPUT);
  pinMode(pinIN3, OUTPUT);
  pinMode(pinIN4, OUTPUT);
  pinMode(pinIN5, OUTPUT);
  //pinMode(pinIN6,); //buzzer r3 entrada// ja esta no controle
  pinMode(pinIN7, OUTPUT); //buzzer saida
}

```

```
52 boolean disparo = false; //Buzzer permanece desativado
53
54 void loop() {
55   //*************** Controle do RF ***********************
56   if (radio.available()) {
57      radio.read( &dadosRF, sizeof(tipoDadosRF) );
58   }
59   Serial.println(digitalRead(dadosRF.pot3));
60
61
62   //*************** Controle do Carro ************
63      //Controle da direção
64      if (dadosRF.pot2 < 512) {
65         //Esquerda
66         pDireita  = 100;
67         pEsquerda = map(dadosRF.pot2, 511, 0, 100, 0);
68      } else {
69         //Direita
70         pDireita  = map(dadosRF.pot2, 512, 1023, 100, 0);
71         pEsquerda = 100;
72      }
73
74      if (dadosRF.pot1 < 512) {
75         //Reverso
76         int velocidade = map(dadosRF.pot1, 511, 0, 0, 255);
77
78         analogWrite(pinIN1, 0);
79         analogWrite(pinIN2, velocidade * pDireita / 100);
80
81         analogWrite(pinIN3, 0);
82         analogWrite(pinIN4, velocidade * pEsquerda / 100);
83
84          //*************** Controle da buzina ***********
85       if(digitalRead(dadosRF.pot3) == LOW) { //botao manete pressionado
86         //disparo = true;   //Buzzer emite som
87         tone(pinIN7, 440);
88       }
89       else {
90         //disparo = false;            //Buzzer não emite som
91         noTone(pinIN7);
92       }
93       /*if(disparo == true) { //Se botao da manete for pressionado, o buzzer
     emite som
94         tone(pinIN7, 440);   //Define a frequência em 440
95         //delay(200);
96       }*/
97
98         //*************** Controle das luzes ************
99           //Mantem LED apagado
100          digitalWrite(pinIN5, LOW);
101     } else {
102        //Para frente
103        int velocidade = map(dadosRF.pot1, 512, 1023, 0, 255);
```

```
104
105         analogWrite(pinIN1, velocidade * pDireita / 100);
106         analogWrite(pinIN2, 0);
107
108         analogWrite(pinIN3, velocidade * pEsquerda / 100);
109         analogWrite(pinIN4, 0);
110
111
112          //*************** Controle da buzina ***********
113       if(digitalRead(dadosRF.pot3) == LOW) { //botao manete pressionado
114          // disparo = true;    //Buzzer emite som
115           digitalWrite(pinIN7, HIGH);
116       }
117       else {
118         //disparo = false;             //Buzzer não emite som
119           digitalWrite(pinIN7, LOW);
120       }
121       //*************** Controle das luzes ************
122       //Mantem LED aceso
123           digitalWrite(pinIN5, HIGH);
124
125     }
126  }
```

Funcionamento do Carrinho de Controle Remoto
O Carrinho de Controle Remoto é alimentado por uma bateria de 7,4V (bateria Lipo de duas células), o Led verde da placa de circuito impresso estará aceso indicando que o microcontrolador ATMEGA328P está recebendo 5V e o Led Vermelho da pont H também estará aceso indicando que o carrinho está ligado e deve estar funcionando.
O Controle é alimentado por uma bateria de 3,7V (bateria lipo de uma célula) de até 300 mAh porque o consumo do controle é muito baixo e acima de 300 mAh a bateria começa a ficar grande e pesada. O Led vermelho interno do Arduino ProMini irá acender indicando que o controle está ligado. É muito importante tomar cuidado ao ligar a bateria do controle, conectando fio vermelho com fio vermelho, ou seja, positivo com positivo e negativo com negativo. Se essa recomendação nao for seguida, há grande chance de danificar o circuito e ser necessário a troca de peças.
O Carrinho é controlado da seguinte forma: o analógico da esquerda controla com o movimento vertical se o carrinho irá para frente ou para trás; o analógico da direita controla com o movimento horizontal se o carrinho ira se direcionar para direita ou para esquerda, porém, essa função só funciona se o carrinho ja estiver indo para frente ou para trás.

III. Resultados

O projeto cumpriu com a proposta de ser um carrinho de controle resistente, mesmo ao passar por diversos impactos continua funcionando. É possível utilizar ele por diversas horas sem precisar ao menos trocar a bateria que se encontra no carrinho. O custo final

do carrinho é barato comparado aos brinquedos do mercado brasileiro, então, vale apena reproduzi-lo.
Por fim, o Carrinho de Controle Remoto passou por diversas aprensentações tendo um excelente resultado e gerando interesse do público.

IV. Discussão

Os projetos do PETEE com comunicação sem fio tem passado por problemas de interferência, sendo assim, o carrinho de controle tem em determinadas situações perdido literalmente o controle por esse tipo de problema. Infelizmente não se acha muito conhecimento em livros e na internet sobre esse assunto, como a aplicação de filtros, a maioria é relatos práticos de aplicação de filtro para eliminação de ruídos de interferência. Esses relatos foram levados em consideração mas não obteve-se resultado satisfatório. Medidas que ja foram implementadas: indutor de núcleo de ferrite na alimentação para tirar os ruídos vindos principalmente dos motores; 3x capacitores de 100nF cerâmicos com ligação do tipo triângulo nos motores; capacitor eletrolítico na alimentação do NRF. Sendo assim, a alternativa para o futuro é procurar professores especializados nessa área para auxiliar na solução deste insistente problema.
Algumas vezes o carrinho está ligado, o controle também, mas ele não funciona. Possíveis problemas: NRF do carrinho desencaixou deivido ao impacto; fios com mal contato na pont H; jumpers soltaram do microcontrolador ATMEGA; o fio da bateria que liga na ponte H pode estar solto; os fios do NRF do controle podem estar com mal contato (os fios sao muito finos e velhos com um pouco de oxidação); foi feito a ligação da bateria do controle com polaridade invertida, podendo ter queimado algum componente. Para evitar que aconteça a ligação errada da bateria do controle é necessário no futuro criar ou comprar um tipo de encaixe de bateria.

Referências

<https://www.youtube.com/watch?v=61M7Mg7KP6k&t=4s>
<https://www.youtube.com/watch?v=ZAKyxNs2uuA&t=1684s>