Curso Básico de Robótica

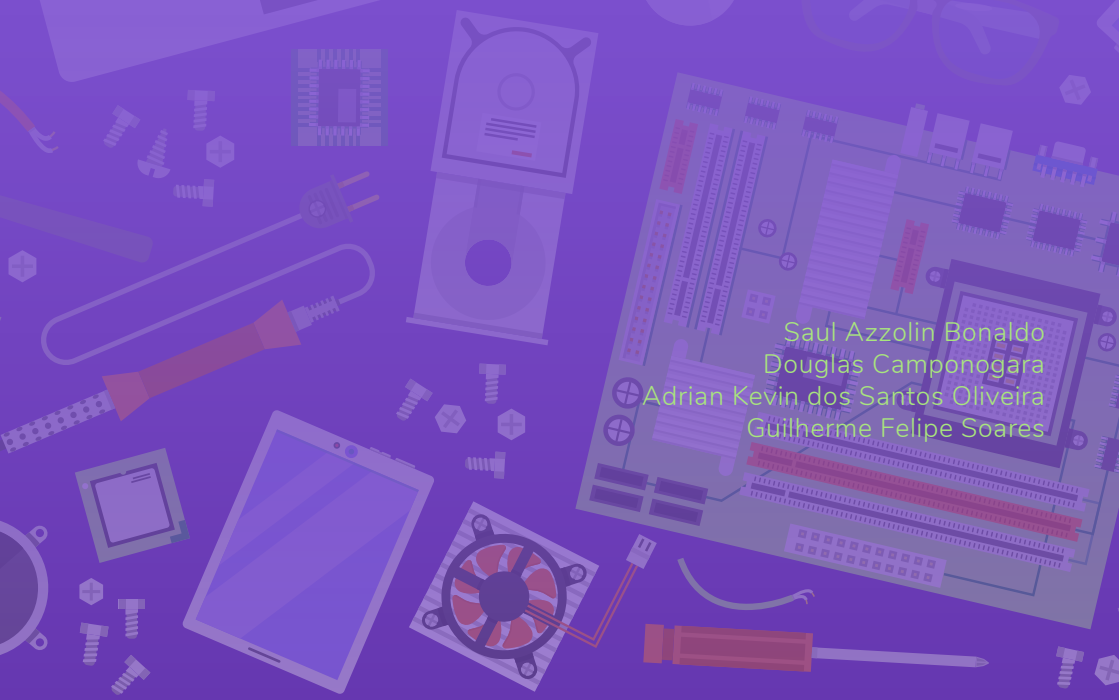
Saul Azzolin Bonaldo
Douglas Camponogara
Adrian Kevin dos Santos Oliveira
Guilherme Felipe Soares

2023

4 EDUCAÇÃO DE QUALIDADE

Este material atende a um ODS da ONU

Curso Básico de Robótica



Saul Azzolin Bonaldo
Douglas Camponogara
Adrian Kevin dos Santos Oliveira
Guilherme Felipe Soares

1.ª Edição

Santa Maria
Pró-Reitoria de Extensão - UFSM
2023

C977 Curso básico de robótica [recurso eletrônico] / Saul Azzolin Bonaldo ... [et al.]. – 1. ed. – Santa Maria/RS : UFSM, Pró-Reitoria de Extensão, 2023.
1 e-book : il. – (Série Extensão)

ISBN 978-65-85653-17-6

1. Robótica - cursos 2. Arduino 3. Sensores 4. Linguagem de programação 5. Atuadores 6. Diagrama de blocos I. Bonaldo, Saul Azzolin

CDU 004.896

Ficha catalográfica elaborada por Lizandra Veleda Arabidian - CRB-10/1492
Biblioteca Central - UFSM

CONSELHO EDITORIAL

CÂMARA DE EXTENSÃO

Flavi Ferreira Lisbôa Filho
Presidente

Jaciele Carina Vidor Sell
Vice-Presidente

José Orion Martins Ribeiro
PROPLAN

Marcia Regina Medeiros Veiga
PROGRAD

Michele Forgiarini Saccol
CCS

Monica Elisa Dias Pons
CCSH

Andre Weissheimer de Borba
CCNE

Suzimary Specht
Politécnico

Marta Rosa Borin
CE

Thiago Farias da Fonseca Pimenta
CEFD

Marcia Henke
CTISM

Adriano Rudi Maixner
CCR

Graciela Rabuske Hendges
CAL

Ana Beatris Souza de Deus Brusa
CT

Tanea Maria Bisognin Garlet
Palmeira das Missões

Fabio Beck
Cachoeira do Sul

Evandro Preuss
Frederico Westphalen

Regis Moreira Reis
TAE

Elisete Kronbauer
TAE

Suélen Ghedini Martinelli
TAE

Isabelle Rossatto Cesa
DCE

Daniel Lucas Balin
DCE

Jadete Barbosa Lampert
Sociedade

PARECERISTA AD HOC

Sidnei Renato Silveira

Cartilha aprovada em sessão ordinária da Câmara de Extensão no dia 17/08/2022.

APRESENTAÇÃO

Este material foi desenvolvido para subsidiar as aulas do projeto de extensão intitulado "CTISM na Educação Básica: Cursos de Robótica na Rede Municipal de Ensino", o qual tem como principal objetivo promover cursos, oficinas e treinamentos em robótica aos docentes e aos alunos das escolas municipais de ensino fundamental de Santa Maria. Para o estudo desta apostila, basta que os discentes tenham conceitos básicos de informática e matemática.

No cenário atual, é perceptível um aumento considerável do interesse pela área da tecnologia, inclusive na eletrônica com o uso de microcontroladores, onde plataformas mais amigáveis como o Arduino tem se destacado. Neste curso abordaremos componentes eletrônicos, o Arduino e, por fim, uma plataforma para as simulações. Os elementos utilizados têm um custo baixo e acessível, assim possibilitando as suas aplicações em brinquedos ou em atividades domésticas simples.

De acordo com Amariei (2015), o Arduino vem se tornando o padrão no desenvolvimento de projetos eletrônicos microcontrolados, pois além de ter um custo baixo, permite que o usuário crie os seus próprios projetos eletrônicos em um tempo menor em relação às plataformas tradicionais.

O Arduino tem crescido entre os *hobbistas*[1] por existir muito conteúdo sobre seu uso, não somente em sua página oficial, mas também na internet, tais como os exemplos de projetos explicados de forma didática e com metodologias que não necessitam de conhecimentos profundos de programação.

Uma dessas metodologias é o uso de interface gráfica, chamada de diagrama de blocos, em que é possível programar sem ter conhecimento das linguagens, através de formas e setas que mostram o fluxo de execução do programa. Existem programas que dispõem desses "blocos" para introduzir as pessoas à linguagem de programação, sendo abordado neste curso o Block.ino.

Espera-se, ao final deste material, que o estudante consiga montar o kit de robótica fornecido às escolas gaúchas por um programa do Estado do Rio Grande do Sul, bem como programar o seu Arduino para a execução de movimentos básicos, leitura de sensores e atuação através de comandos enviados pelo celular.

[1] Hobbista é um termo em inglês que se refere a uma pessoa que se dedica, de forma prazerosa, a uma determinada tarefa.

SUMÁRIO

COMPONENTES, ARDUINO E PROGRAMAÇÃO

1

Neste capítulo, abordaremos o conteúdo necessário para que o aluno possa embarcar nos estudos da eletrônica, entendendo como funciona cada componente eletrônico e a plataforma Arduino, além de trabalhar com o Block.ino.

1.1 COMPONENTES DO KIT DE ROBÓTICA

Nesta seção apresentaremos os componentes eletrônicos fornecidos no kit de robótica, bem como o seu princípio e funcionamento.

1.1.1 LED

O Diodo Emissor de Luz (*Light Emitting Diode* - LED) é um dos componentes mais utilizados na eletrônica básica, sendo capaz de emitir luz quando uma corrente elétrica flui pelos seus terminais. O LED é visto em diversas aplicações, como: semáforos, lâmpadas, televisores, entre outras aplicações.

Os LEDs podem ser encontrados de diversas cores, como em luzes decorativas e os chamados "pisca-pisca". Também existem os LEDs RGB, das siglas *Red* (vermelho), *Green* (verde), *Blue* (azul), que permitem alternar entre

essas três cores e também misturá-las, possibilitando a criação de outros tons como, por exemplo, a cor branca que seria a união delas.

A aplicação dos LEDs na placa Arduino é relativamente simples, bastando apenas aplicar um sinal de tensão. Porém para que não haja nenhum problema na hora de ligar o nosso LED, é imprescindível que o usuário conecte um resistor no terminal positivo do LED (o Ânodo) para limitar a energia excedente que sai da placa para o LED, e em seguida conecte o mesmo a algum pino de Entrada/Saída digital. Para vermos os valores corretos dos resistores podemos consultar a Figura 1:

Figura 1 - Resistências recomendadas

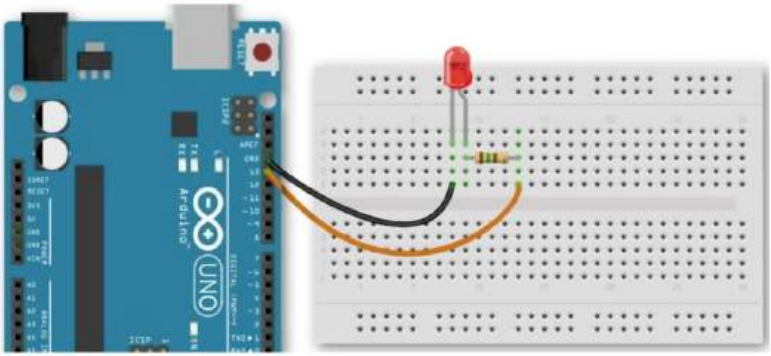
LED's	Tensões	Resistores	
		3.3V	5V
	2.2V	33 Ω	150 Ω
	2.2V	33 Ω	150 Ω
	2.2V	33 Ω	150 Ω
	3.4V	0 Ω	82 Ω
	3.4V	0 Ω	82 Ω
	3.4V	0 Ω	82 Ω
	3.4V	0 Ω	82 Ω

Fonte: Autor.

Como é possível observar na Figura 1, dependendo da cor do LED o valor da resistência em série muda. Além disso, também é fundamental observar o valor da tensão aplicada no LED. No caso de aplicações da placa Arduino, utilizaremos apenas a última coluna, pois a maioria dos modelos de placas da família Arduino trabalham com

sinais lógicos de 5V. Após determinarmos o valor do nosso resistor para limitar a corrente, devemos ligar o terminal negativo do LED (Cátodo) ao nosso pino GND, conforme a Figura 2. Por fim, podemos fazer nosso código de preferência do usuário, para acender o LED basta o usuário definir o LED como saída e marcar como ALTO para ligar o LED, e para desligar o LED basta sinalizar o LED como BAIXO.

Figura 2 – Circuito para ligar um LED



Fonte: Autor.

1.1.2 Sensor Ultrassônico

O sensor ultrassônico (mostrado na Figura 3) funciona por meio da emissão de ondas sonoras de alta frequência, sendo que esta ondulação é refletida quando atinge algum objeto. O tempo em que o sinal emitido demora para retornar ao sensor tem relação direta com a distância entre o objeto e o sensor. Os sensores ultrassônicos são capazes de identificar a posição de objeto, contar objetos, estimar distâncias, entre outras diversas aplicações.

Figura 3 – Sensor ultrassônico

Fonte: Autor.

Como explicado anteriormente, o lado direito do sensor emite uma onda sonora de alta frequência e, quando rebatido a um objeto, o lado esquerdo do sensor consegue receber um tipo de eco na forma de sinais elétricos.

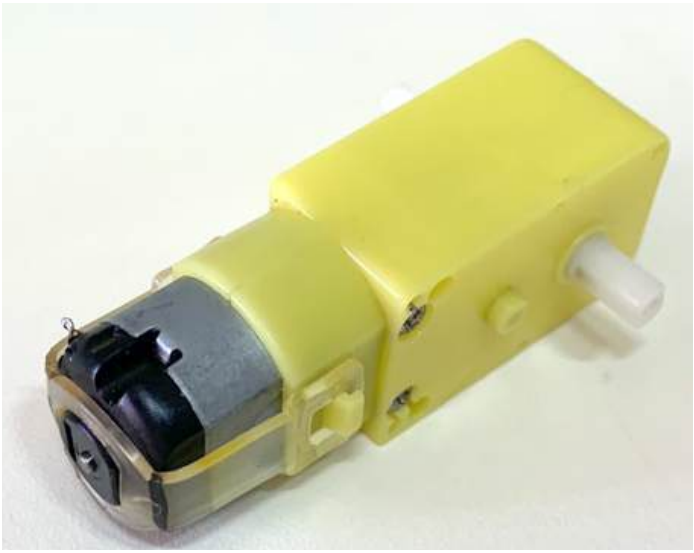
1.1.3 Motores

Os motores são componentes capazes de transformar energia elétrica em energia mecânica. Existem dois tipos de motores: os motores de Corrente Contínua (CC) e os motores de Corrente Alternada (CA). A corrente alternada, refere-se à energia proveniente da nossa rede elétrica, por exemplo, a tomada de nossa casa, a qual possui uma energia que está em constante variação. Já a corrente contínua, refere-se a uma energia sempre contínua, sem quedas ou variações.

Para trabalharmos com motores no Arduino, utilizaremos motores CC, como o mostrado na Figura 4. Explicando de forma resumida o funcionamento dos

motores CC, eles são constituídos por dois ímãs, assim gerando um campo magnético entre eles. Também possuem uma bobina com fios de cobre, responsável por interagir com o campo dos imãs permanentes. Por fim, essa energia dentro do campo magnético produz uma força magnética, que passa pelo eixo do motor e gera o torque fazendo com que o nosso motor gire.

Figura 4 – Motor CC



Fonte: Autor.

Para acionar o motor através do Arduino, basta conectar um dos terminais no negativo da placa [ponto esse também chamado de Terra ou *Ground* (GND)] e o outro terminal em algum pino de Entrada/Saída digital da placa do Arduino, e em nosso código definir o pino escolhido com ALTO para girar o motor, e para parar de girar basta defini-lo como BAIXO.

1.1.4 Servomotor

O servomotor (mostrado na Figura 5) funciona da mesma forma que o motor CC, porém para que ele gire é necessário enviar um sinal de controle em nosso programa. Esse sinal determina de maneira precisa até que ponto o motor deve girar, diferentemente dos motores CC, não precisamos ligá-lo a um pino Modulador de Pulso (*Pulse Width Modulator* - PWM). O servomotor tem uma aparência semelhante ao do motor CC, porém possui uma hélice maior, possibilitando a conexão com outros dispositivos, como o sensor ultrassônico. É bastante empregado em movimentos precisos, como o braço de um robô, por exemplo.

Figura 5 – Servomotor

Fonte: Autor.

O servomotor possui três pinos de ligação, sendo um deles o de "sinal", o qual deve ser ligado em algum pino de Entrada/Saída digital da nossa placa, um pino de "potência", o qual conecta-se nos 5 V, e um pino "GND", que deve ser ligado no GND de nossa placa Arduino. Por fim, no programa do nosso servomotor, basta o usuário escolher quantos graus ele quer que o servo gire.

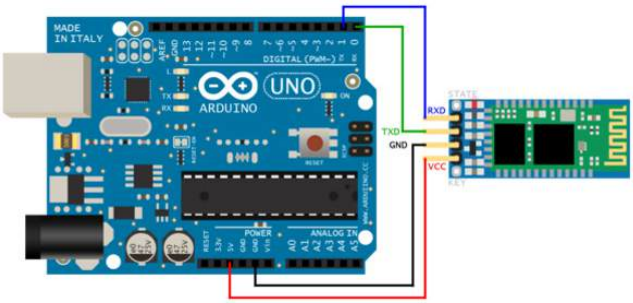
1.1.5 Bluetooth

O Bluetooth é uma forma de conectar, sem fios, diversos tipos de aparelhos, sendo muito utilizado pelos usuários para enviar arquivos para outros celulares e ligar dispositivos que possuam esse tipo de conexão.
Para conectarmos o Bluetooth ao Arduino, é necessário utilizar uma placa, chamada de Módulo Bluetooth (demonstrado na Figura 6), a qual possibilita que o usuário receba e transmita dados por meio da comunicação sem fio. Por exemplo, o usuário pode ligar um LED enviando dados do celular para o módulo ligado à placa Arduino.

Os módulos trabalham na função mestre, que consegue parear[2] com qualquer dispositivo Bluetooth e enviar dados, como por exemplo um sensor de temperatura, e receber estes dados de temperatura no celular pareado. Também trabalham na função escravo, onde eles apenas aceitam o pareamento e podem receber dados, como, por exemplo, receber comandos de um celular para controlar algum componente.

[2] Parear é um processo que permite estabelecer uma relação de confiança entre dois dispositivos Bluetooth, através da troca de informações de segurança. Sem ele, o seu celular (ou outros dispositivos) poderia se conectar aleatoriamente com qualquer outro dispositivo.

Figura 6 – Módulo Bluetooth conectado à placa Arduino



Fonte: Autor.

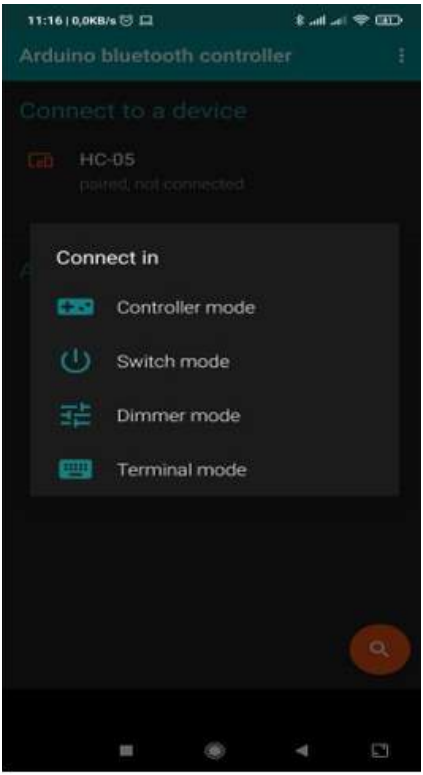
Na Figura 6 vemos como é feita a ligação do módulo Bluetooth. Para ligarmos algum componente, devemos ter um celular em mãos, seja ele de sistema Android ou iOS, nele o usuário deve fazer o download de um controlador Bluetooth, que está disponível tanto na Play Store (para usuários de Android) quanto na Apple Store (para usuários de iOS).

Quando ligamos o nosso módulo já tendo o código feito (que será explicado no Capítulo 2 - Montagem e funcionamento básico), ele se torna pareável. Logo, quando abrirmos o aplicativo já poderemos ver o nosso módulo visível para conectar. Como mostrado na Figura 7, o aplicativo fornece 4 formas de controle:

- *Controller mode:* é um controle de videogame simples (um Joystick);
- *Switch mode:* é um interruptor onde podemos ligar e desligar algo;

- *Dimmer mode:* é uma forma de controle onde podemos controlar o volume, intensidade, ou velocidade de algo;
- *Terminal mode:* um teclado onde podemos digitar o que quisermos para enviar para nosso módulo.

Figura 7 – Conectando o módulo Bluetooth no aplicativo



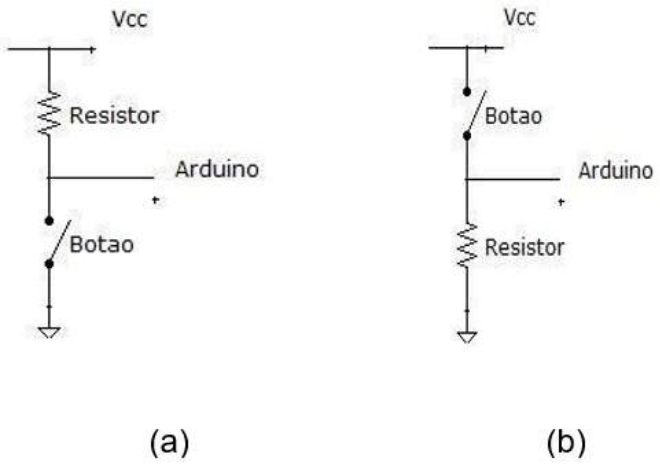
Fonte: Autor.

1.1.6 Botões

Os botões são componentes indispensáveis na eletrônica, eles servem para ligar circuitos, fazer alteração de valores, entre outras diversas aplicações, como podemos vivenciar em nosso dia-a-dia. Muitos dos objetos que utilizamos em nossas residências possuem um sistema com botões, tais como o telefone fixo, controle remoto, interruptor para sistemas de iluminação, entre outros.

Quando ligamos um botão em algum circuito, nele ocorrem variações de tensão logo após pressioná-lo. Para evitar essas variações devemos utilizar resistores, os quais são chamados de resistores *pull-down* e resistores *pull-up*, conforme mostra a Figura 8.

Figura 8 – Botões com resistores: (a) *pull-up* e (b) *pull-down*



Fonte: Autor.

Na Figura 8 podemos ver detalhadamente como funciona a ligação dos botões com resistores *pull-up* e com resistores *pull-down*. No caso de pull-up, ao ligarmos o circuito, a placa receberá um sinal ALTO e irá se manter ligada. E, quando pressionamos o botão, a placa irá receber um sinal BAIXO, ligando o botão diretamente ao GND, fazendo com que ele desligue o circuito. Nas aplicações do Arduino, os pinos de Entrada/Saída digital possuem um resistor *pull-up* integrado que pode ser habilitado por software, não necessitando desta forma a colocação de um resistor no circuito.

Já no caso de *pull-down*, ocorre o inverso: quando ligarmos o circuito, ele se mantém em nível lógico BAIXO, desligado, e quando pressionamos o botão nosso circuito liga, recebendo o sinal ALTO de 5 V. Essas ligações servem para que não haja variação de tensão logo após o botão ser pressionado. Por fim, no seguinte link há um vídeo com a explicação de cada componente citado anteriormente.

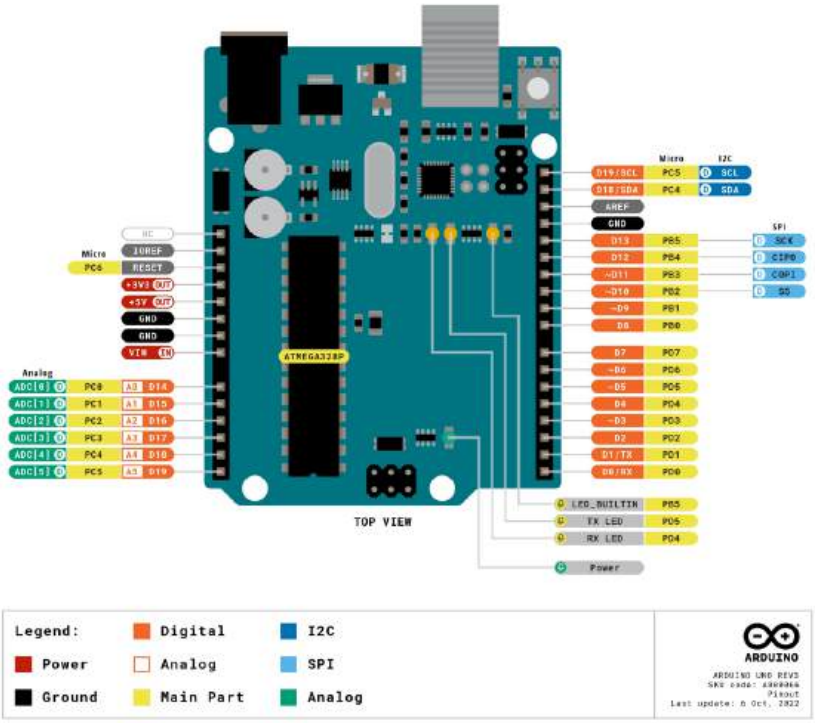
1.2 ARDUINO

O Arduino possui uma gama de utilidades, além de ser uma plataforma de baixo custo. Ele acompanha um cabo com conexão do tipo *Universal Serial Bus* (USB) para conectar ao seu computador, permitindo assim o acesso ao seu software, possibilitando a programação.

O Arduino UNO (mostrado na Figura 9) possui 14 pinos digitais de Entrada/Saída,6 deles podendo ser usados como saídas PWM, e os outros 6 pinos de entrada para sinais

analógicos e os pinos de alimentação. Podemos visualizar melhor a sua pinagem, por meio de um *datasheet*[3].

Figura 9 – Diagrama com a pinagem do Arduino



Fonte: Arduino Documentation, 2023. Disponível em: https://docs.arduino.cc/hardware/uno-rev3.

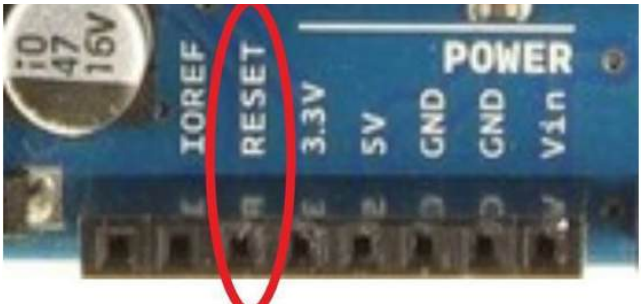
[3] Datasheet é um termo em inglês que significa folha de especificação. Na área eletrônica, o datasheet de qualquer componente funciona como um manual de instrução muito detalhado.

1.2.1 Pinos POWER

Os pinos especificados na categoria POWER são essenciais tanto para fornecimento de energia aos componentes periféricos do Arduino quanto para a verificação do bom funcionamento da placa. Seguem abaixo explicações detalhadas sobre a funcionalidade de cada um desses pinos.

RESET: Mostrado na Figura 10, este pino possibilita que o usuário reinicie sua placa Arduino.

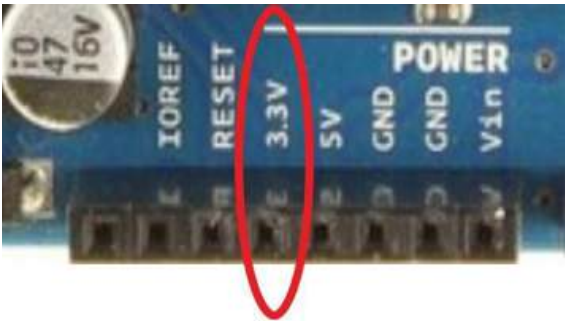
Figura 10 – Pino de Reset



Fonte: Autor.

3.3V: Mostrado na Figura 11, este pino fornece uma tensão de 3,3 V e uma corrente máxima de 50mA.

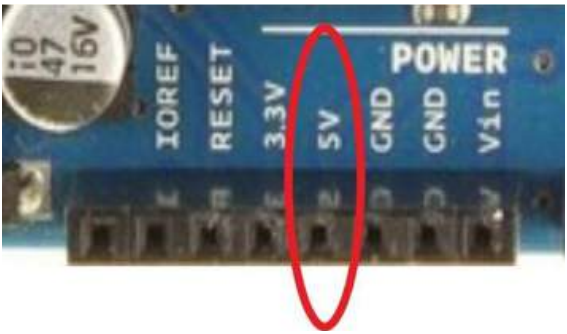
Figura 11 – Pino de 3,3 V



Fonte: Autor.

5V: Mostrado na Figura 12, este pino fornece uma tensão 5 V e uma corrente máxima de 40 mA.

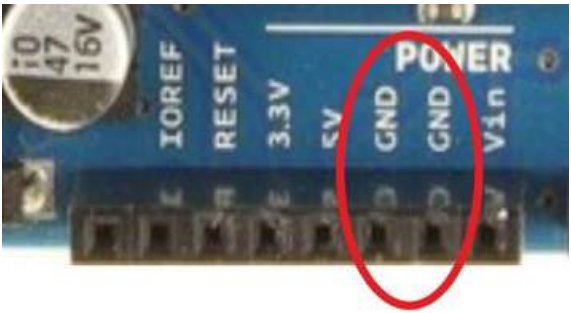
Figura 12 – Pino de 5 V



Fonte: Autor.

GND: Mostrado na Figura 13, esse pino serve como referência de um ponto neutro.

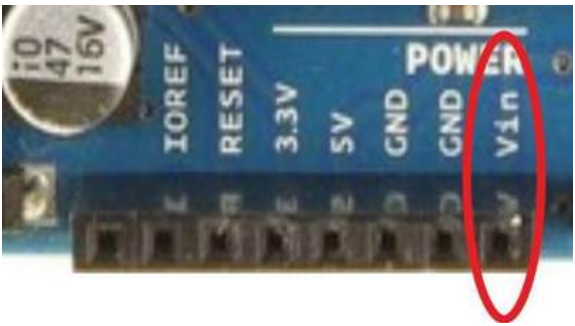
Figura 13 – Pino GND



Fonte: Autor.

Vin: Mostrado na Figura 14, esse pino é utilizado para alimentar o Arduino com uma bateria ou fonte externa.

Figura 14 – Pino Vin



Fonte: Autor.

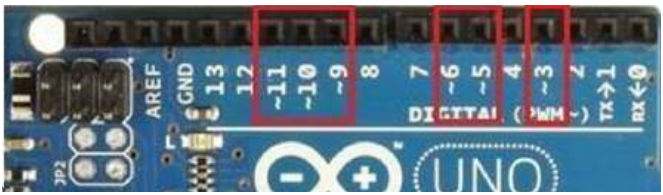
1.2.2 Pinos Digitais (PWM~)

Os pinos digitais podem atuar como Entrada/Saída (*Input/Output*), ou seja, eles podem receber um sinal de um componente ou mandar esse sinal para este mesmo componente. Além disso, os pinos marcados com um til (~) podem realizar uma função adicional, que é a de fornecer um sinal do tipo PWM (Pulse Width Modulation - modulação por largura de pulso).

Exemplo: se um usuário quiser acender e apagar um LED por meio de um botão, ele teria que ligar o LED a um pino digital e defini-lo em seu programa como um pino de saída (*output*). Já para o botão, o mesmo deve ser ligado a outro pino digital, sendo esse definido como entrada (*input*). Esses pinos fornecem uma tensão de 5 V, trabalhando apenas com dois níveis de tensão: 0V (BAIXO) ou 5V (ALTO).

Neste vídeo, a placa Arduino é apresentada, bem como suas funcionalidades. Na placa Arduino Uno encontramos 6 portas digitais que podem ser utilizadas como sinal de saída PWM (mostrado na Figura 15). As portas PWM são muito utilizadas para fazer o controle da velocidade de motores, controle de iluminação, geração de sinais analógicos, sendo também empregadas para gerar sinais de áudio.

Figura 15 – Pinos PWM na placa Arduino

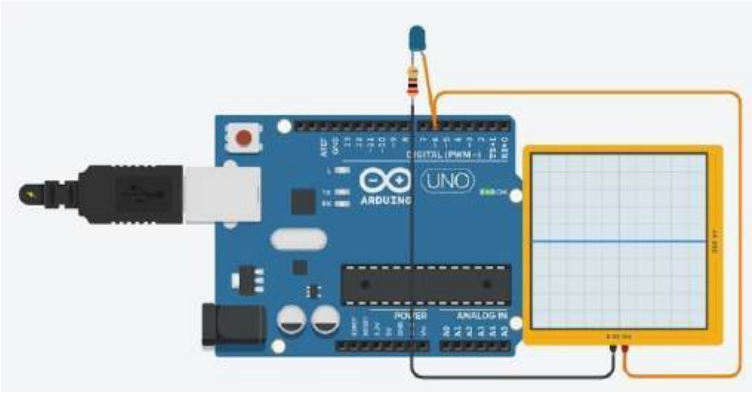


Fonte: Autor.

Os pinos PWM são empregados para controlar dispositivos de maneira mais precisa. O sinal desses pinos alterna entre um valor BAIXO e um valor ALTO com uma frequência fixa. Porém, o tempo em que a tensão permanece no valor ALTO muda conforme a necessidade. Quanto maior a velocidade ou a potência a ser enviada para a carga, maior será o tempo em que o sinal ficará em valor ALTO.

Para uma melhor demonstração do PWM, será utilizado um LED para mostrar a intensidade do brilho e um osciloscópio onde podemos ver a onda do sinal elétrico que está sendo gerado. Na Figura 16 temos um PWM em que a tensão não está presente na saída, logo o LED está apagado.

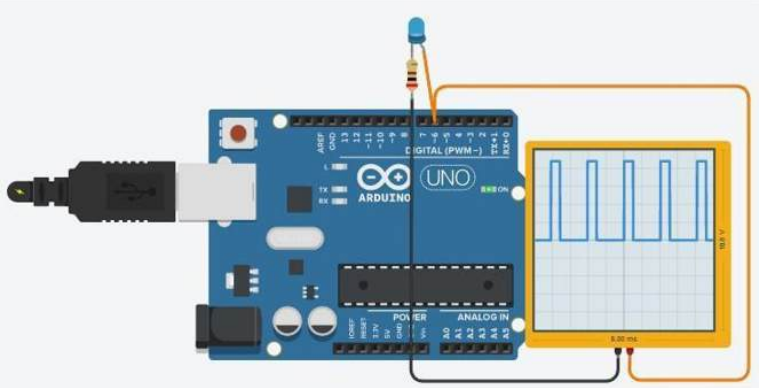
Figura 16 – PWM em 0% de ciclo de trabalho



Fonte: Autor.

Na Figura 17, a tensão de saída está 25% ligada e 75% desligada, ou seja, temos um ciclo de trabalho de 25%. O LED começa a brilhar um pouco.

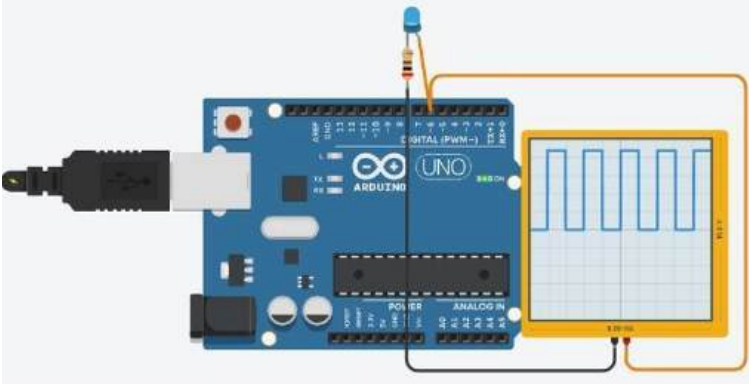
Figura 17 – PWM em 25% de ciclo de trabalho



Fonte: Autor.

Já na Figura 18, temos um ciclo de trabalho de 50% no nosso sinal PWM. É possível perceber que o LED está aceso e com mais brilho.

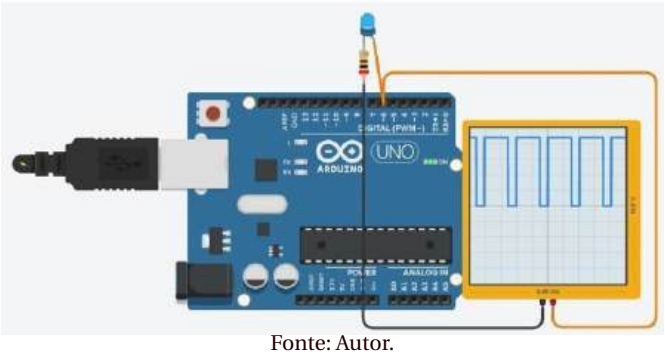
Figura 18 – PWM em 50% de ciclo de trabalho



Fonte: Autor.

Na Figura 19, com 75% de ciclo de trabalho, já é possível ver um brilho mais nítido em nosso LED, porém ainda não é o seu máximo. Na imagem do osciloscópio pode-se perceber que o nosso sinal está 75% do tempo em nível lógico alto e 25% em nível lógico baixo.

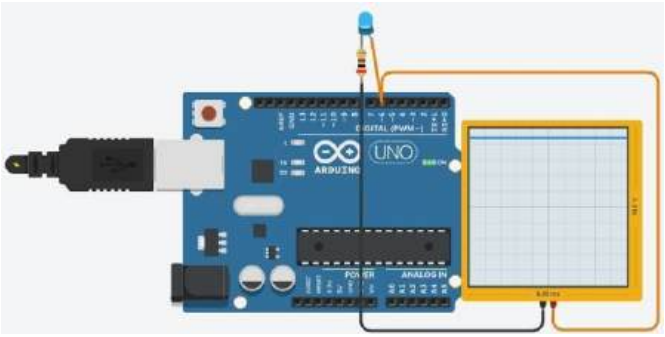
Figura 19 – PWM em 75% de ciclo de trabalho



Fonte: Autor.

Neste último exemplo, apresentado na Figura 20, demonstramos o nosso sinal PWM em 100% de ciclo de trabalho, onde podemos ver o LED aceso com seu brilho máximo e a tensão de entrada completamente aplicada na saída.

Figura 20 – PWM em 100% de ciclo de trabalho



Fonte: Autor.

Também é importante saber que o ciclo de trabalho dos pinos PWM da placa Arduino UNO é configurado com valores 0 a 255, ou seja, em 0 temos um ciclo de trabalho de 0%, com 127 tem-se 50% de ciclo de trabalho, e com 255 temos, por fim, 100%.

1.3 CONCEITOS BÁSICOS DE PROGRAMAÇÃO

Para fazer com que tudo já citado anteriormente funcione, devemos ter em mente que a plataforma Arduino é totalmente programável. Logo, é necessário escrever um código que configure e idealize o funcionamento de cada componente, onde definimos o que cada um deve fazer, e o que cada um é. E então o computador tem a função de ler este código, traduzi-lo para código binário e enviá-lo para o Arduino.

1.3.1 Algoritmo

Em tudo o que fazemos em nossa rotina diária mantemos uma ordem de tarefas, como, por exemplo, acordar, tomar café, estudar, almoçar, trabalhar, jantar e dormir, e isso se repete durante os dias. Dentro da programação também devemos ter uma ordem em nossas ações para que elas sejam executadas de forma correta, sendo esse roteiro de ações chamado de algoritmo.

1.3.2 Linguagens de programação

Existem vários tipos de linguagens de programação atualmente. As linguagens de programação fazem a ponte entre a nossa maneira de escrever uma rotina de ações e a forma como a máquina entende. O código de programação não é algo muito visto diariamente, podendo à primeira vista parecer complexo e de difícil entendimento. Porém, com a linguagem de programação do Arduino, este ambiente programável se torna muito mais simples se comparada a outras linguagens de programação já existentes.

Para piscarmos um LED, por exemplo, basta criarmos uma função de configuração, no caso o *void setup*(), onde dentro desta função determinamos o pino 13 (onde foi ligado o LED) como um sinal de saída (*OUTPUT*) para o LED. Logo após criamos outra função que seria o *void loop*(), esta função faz com que a placa repita os comandos até que seja desligada. Em resumo, estas duas funções são a estrutura básica da programação do Arduino, que contém uma função para configuração dos componentes [*void setup*()], e outra para repetir os comandos desejados [*void loop*()] até que a placa seja desligada.

1.4 PROGRAMANDO

Quando um programa é executado em um Arduino, é possível verificar o quão interessante ele é capaz de se tornar, além de ser muito gratificante ver um algoritmo que nós mesmos fizemos ser executado da forma que desejamos. Para o usuário aprender a programar, existem muitos programas e jogos divertidos e didáticos para a programação, como, por exemplo, o Scratch. Este software é gratuito e mostra aos usuários a programação de uma forma bem simples. Também existem outros programas de fácil uso e acesso que introduzem muito bem a programação. Neste material, para facilitar a construção do algoritmo para Arduino, utilizaremos o Block.ino.

1.4.1 Block.ino

O Block.ino é um *software* que utiliza o princípio de programação em blocos, fornecendo uma programação ilimitada. Para facilitar ainda mais, foi desenvolvido pela equipe Nightwind um software baseado no Block.ino que apresenta as mesmas características, porém foi modificado para atender a demandas do Kit de Robótica distribuído para as escolas municipais gaúchas. Este software desenvolvido pela equipe Nightwind foi denominado de *NightwindBlockly*.

1.4.2 Guia de instalação NightwindBlockly

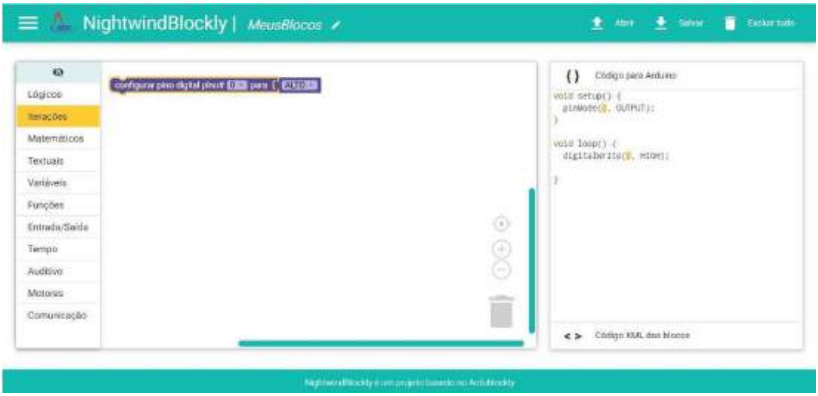
Para abrir o software, é necessário baixar dois arquivos para instalar em seu computador. Foram desenvolvidos vídeos instrucionais demonstrando como se instala este software para ser executado em sua máquina:

- para o sistema Windows: https://youtu.be/c3EeelNl0nE
- para o sistema Linux: https://youtu.be/joBxdxki2XQ

1.4.3 Ambiente de programação do NightWindBlockly

No NightWindBlockly, quando acessamos o programa é possível visualizar os blocos de programação (funções que iremos utilizar), enquanto que no lado direito da tela é transcrito o mesmo código em texto para demonstrar ao usuário, como mostra a Figura 21.

Figura 21 – Exemplo de código blocos+texto

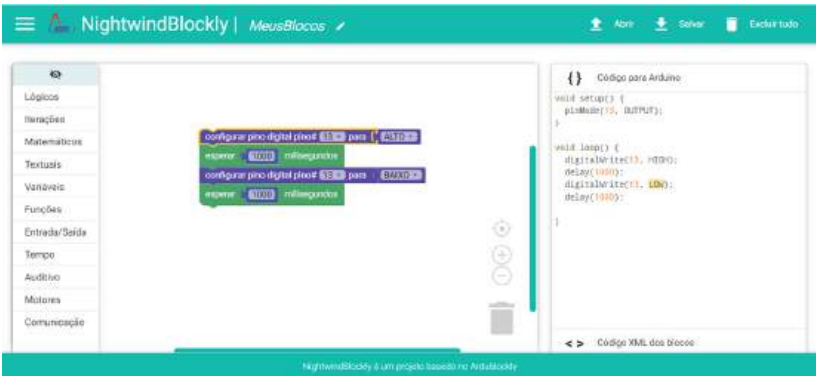


Fonte: Autor.

Para introduzir esta lógica de programação, nós utilizamos a linguagem por blocos, tornando assim o desenvolvimento do código do algoritmo mais intuitivo. A linguagem de programação em blocos tem o mesmo princípio da programação em texto, porém ela utiliza mais a questão lógica, tornando o código de programação mais simples. Como o próprio nome já diz, a programação em blocos se baseia em blocos coloridos, onde cada cor representa sua função, bastando encaixar os blocos para fazer o seu código.

Os blocos roxos são os dados de saída utilizados no circuito. No exemplo mostrado na Figura 22 definimos o LED, que foi atribuído ao pino 13, como estando em estado lógico ALTO (LED Ligado). Em seguida, se encaixa o bloco verde (de tempo) para aguardar 1 segundo. Logo após, definimos o mesmo pino do LED como BAIXO (LED desligado) e aguardamos mais 1 segundo, para após retornar ao início do laço de repetição, repetindo desta forma esses comandos, este seria nossa função de repetição (representado na programação em texto por *void loop*(), e a nossa função de configuração *void setup*() já está embutida nos próprios blocos que já são definidos.

Figura 22 - Exemplo de código para piscar um led ligado ao pino 13

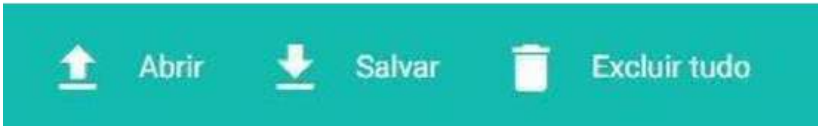


Fonte: Autor.

1.4.4 Gravando um programa no Arduino

Após abrir o ambiente NightwindBlockly e programar o seu diagrama de blocos, é necessário salvar este código para que possamos inseri-lo no ambiente de programação do Arduino para executar em sua placa. Logo, basta clicar em salvar no NightwindBlockly, conforme mostra a Figura 23.

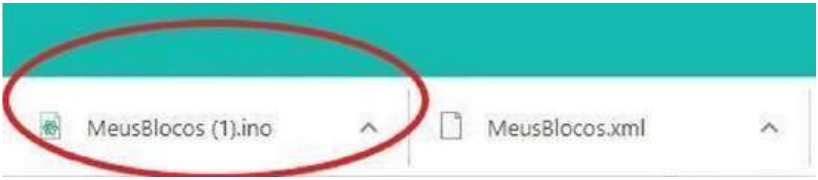
Figura 23 - Botão Salvar na barra de menu do NightwindBlockly



Fonte: Autor.

Quando salvo, o software irá baixar dois arquivos, conforme mostra a Figura 24. O arquivo ".ino" será o qual devemos abrir no software do Arduino.

Figura 24 - Arquivos baixados do NightwindBlockly



Fonte: Autor.

Para demonstrar este processo, foi feito um vídeo mostrando o passo a passo de como gravar o código do NightwindBlockly para o Arduino.

MONTAGEM E FUNCIONAMENTO BÁSICO 2

Na parte prática deste material, mostraremos o passo a passo da montagem de um pequeno robô com o Arduino e o kit de robótica recebido pelas escolas gaúchas. Neste capítulo vamos introduzir os alunos às partes mais básicas da montagem, ou seja, como fazer nosso robô andar para frente, para trás e girar. Para isso utilizaremos bastante o acionamento de motores, o qual será nosso foco neste capítulo.

2.1 MONTAGEM MECÂNICA

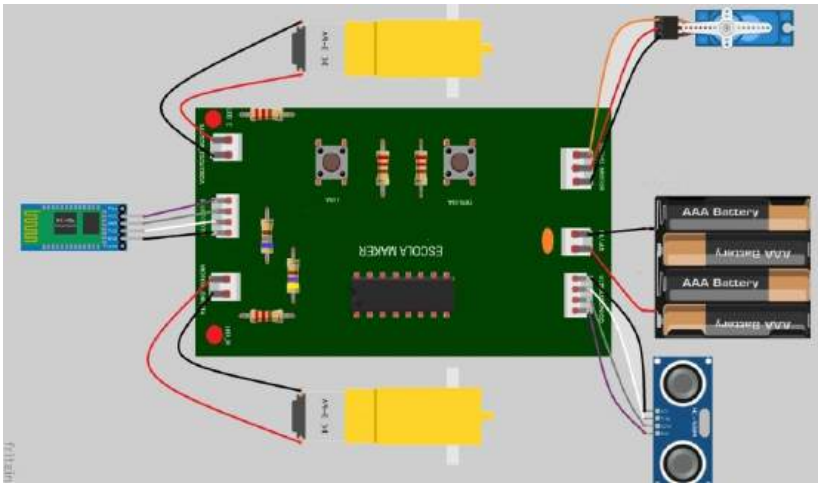
O primeiro passo para a montagem do robô é unir as peças de acrílico que compõem o kit de robótica, formando a estrutura que vai sustentar o seu funcionamento. Cabe ressaltar que as peças são bem delicadas, logo é necessário ter um cuidado ao fazer o encaixe das peças. Demonstramos em um vídeo o passo a passo do processo de montagem mecânica.

2.2 MONTAGEM ELÉTRICA

Para que os motores possam funcionar nesta etapa, eles precisam ser alimentados com um circuito dedicado a este processo. Tal circuito recebe comandos do nosso microcontrolador, o Arduino. A Figura 25

mostra o esquemático dessas ligações. Foi criado um vídeo demonstrando também o passo a passo das conexões elétricas. Um ponto interessante a ressaltar é que a maioria (não a totalidade) dos cabos só podem ser inseridos de uma forma, o que evita conexões invertidas.

Figura 25 – Esquemático da montagem elétrica



Fonte: Autor.

2.3 PROGRAMANDO

Nesta seção vamos programar o cérebro do nosso robô, o Arduino. É por meio dele que será estabelecida a lógica de funcionamento dos componentes. Por isso mostraremos os blocos que devemos programar para o robô, começando pelas funções mais básicas, como fazer funcionar as rodas corretamente: andar para frente e para trás, parar e dobrar. Demonstraremos estas funções em blocos, para que possam ser reproduzidas no

software NightWindBlockly, o seguinte vídeo demonstra como o usuário vai posicionar essas funções no ambiente do Arduino. Nas próximas subseções serão explicadas cada uma dessas funções.

2.3.1 Movimento da roda direita

Para movimentar a roda direita devemos criar três funções: rodadireita_frente, rodadireita_re, rodadireita_parar. Basta irmos em "funções" e escolher a segunda opção "para (faz algo)", em seguida trocamos o nome para a função que desejamos.

Os blocos para rodadireita_frente() e rodadireita_re() são mostrados na Figura 26. Nota-se que em ambos o pino 6 é acionado, pois este funciona como a ignição da nossa roda. Ao contrário de um carro comum, o nosso robô tem um motor para cada roda. Já os pinos 7 e 8 funcionam como um comando de frente e ré. Quando 7 está alto e 8 está baixo, a roda direita vai para frente. Já para ir de ré, basta inverter o comando.

Figura 26 – Diagrama da roda direita: ré e frente

Fonte: Autor.

Por fim, a função rodadireita_parar() é mostrada na Figura 27. Esta função, como o nome diz, imobiliza a roda direita, Neste caso, colocamos tanto o pino 7 quanto o 8 em estado baixo. Este vídeo demonstra esse processo.

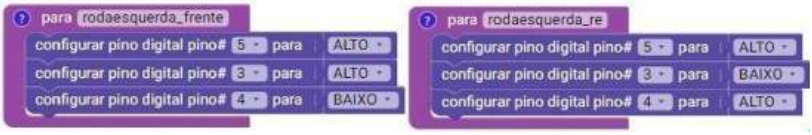
Figura 27 – Diagrama da roda direita: parar

Fonte: Autor.

2.1.2 Movimento da roda esquerda

Para movimentarmos a roda esquerda utilizamos exatamente o mesmo princípio da roda direita apresentada anteriormente, apenas devemos mudar os pinos, que são agora os pinos 5, 4 e 3, como mostra a Figura 28. Desta vez, o pino 5 funcionará como a ignição da roda, e os pinos 4 e 3 como comando de frente e ré. Quando 3 está alto e 4 está baixo, a roda direita vai para frente, já para ir de ré, basta inverter o comando.

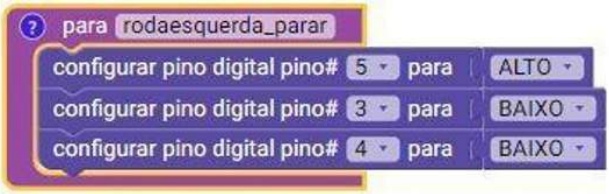
Figura 28 – Diagrama da roda esquerda: ré e frente



Fonte: Autor.

Por fim, a função rodaesquerda_parar() é mostrada na Figura 29. Neste caso, colocamos tanto o pino 4 quanto o 3 em estado baixo.

Figura 29 – Diagrama da roda esquerda: parar



Fonte: Autor.

2.1.3 Movimento completo

Para que possamos movimentar o carrinho para frente e para trás, devemos ligar as duas rodas simultaneamente. Para isso, devemos utilizar as funções apresentadas anteriormente (movimento da roda esquerda e movimento da roda direita). Então devemos criar outro diagrama onde irá se repetir em um loop as nossas instruções, conforme apresentado na Figura 30.

Figura 30 – Fazendo o carrinho andar para frente por 2 segundos e parar

Fonte: Autor.

É possível notar que, após criarmos uma função, o programa nos apresenta ela em um único bloco, assim utilizamos apenas um bloco para determinar se a roda vai para frente, para trás ou se irá parar. No exemplo da Figura 29 nosso carrinho irá para frente e após andar 2 segundos ele irá parar.

Um detalhe importantíssimo é que, para que nosso programa funcione corretamente, as funções apresentadas anteriormente em 2.3.1 e 2.3.2 devem estar abaixo de nosso código.

Na Figura 31 mostramos um exemplo onde o carrinho anda para trás por dois segundos e para, ou seja, o inverso do primeiro exemplo, assim demonstrando claramente que com estas funções prontas se torna muito simples para o usuário movimentar o seu carrinho.

Figura 31 – Fazendo o carrinho andar para trás por 2 segundos e parar

Fonte: Autor.

No exemplo da Figura 30 apenas trocamos os comandos das rodas para que elas girem para trás. Logo, se o usuário quiser que o carrinho ande para frente e para trás, basta empregar as funções em nossos blocos. E, caso o usuário queira que o carrinho ande mais tempo, basta aumentar o tempo do seu código.

Por fim, para que o usuário possa movimentar o carrinho para os lados, basta parar apenas uma das rodas e andar com a outra para frente, ou seja, para movimentar para a esquerda deve-se parar apenas a roda esquerda, para movimentar para a direita deve-se parar apenas a roda direita, conforme pode ser visualizado na Figura 32.

Figura 32 – Fazendo o carrinho andar para frente por 3 segundos, virar para esquerda e continuar andando para frente

Fonte: Autor.

Neste exemplo, o carrinho anda para frente por 3 segundos, vira para a esquerda e segue andando para frente. Percebe-se que foi utilizado um tempo de 500 milissegundos, este é um tempo necessário para que o carrinho vire. Caso este tempo seja maior, o carrinho irá ficar girando e terá outras direções dependentes do tempo empregado. Em um vídeo mostramos um exemplo de movimento completo do nosso carrinho. Nele será apresentado um código exemplo e seu funcionamento.

CONTROLANDO VIA BLUETOOH 3

O Bluetooth, como apresentado na seção 1.1, servirá como um componente de comando para nosso carrinho, ou seja, é por meio do módulo bluetooth que ordenamos que o robô ande para frente, para trás e gire. Com isso podemos gravar um único código para nosso carrinho, com todos os comandos apresentados anteriormente, assim não precisando gravar um comando novo sempre que quisermos trocar a sua direção.

3.1 PROGRAMANDO BLUETOOH

Para a programação do módulo, não será empregado muito além do que o visto anteriormente, apenas utilizaremos algumas funções a mais. Essas funções na verdade se caracterizam como comparações dentro dos blocos, ou seja, definimos se o sinal é maior, menor ou igual a algum caracter. Desta forma, a placa irá entender as suas respectivas funções e poderá receber o sinal estipulado pelo aplicativo para realizá-las dentro dos blocos. Para a construção do nosso código utilizaremos alguns blocos diferentes do que vimos anteriormente, como:

- **Blocos "SE, FAÇA"**: Estes blocos tem a função de fazer algo se o comando que estiver dentro dele for verdadeiro, exemplo: "SE" bluetooth "F", "FAÇA" carrinho andar para a frente. Nota-se que é utilizado um "F" apenas, pois se trata de um comando utilizando um caractere, neste caso o "F" representa "Frente";

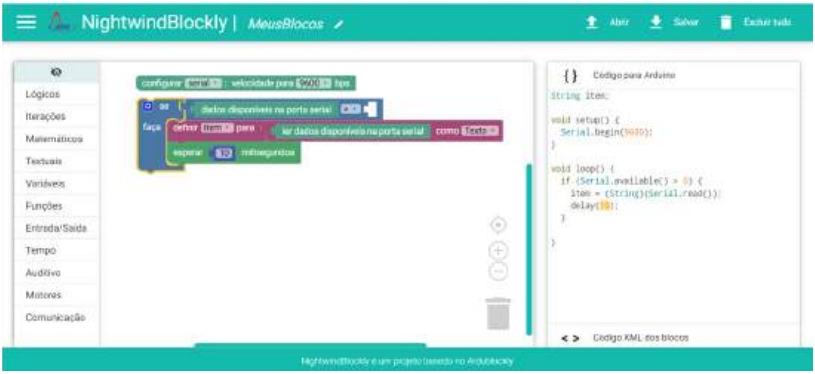
- **Blocos "Variáveis"**: A variável torna possível armazenar valores no programa, tais como: informações, cálculos e dados de sensores;

- **Blocos "Comunicação"**: Estes blocos servem para que seja possível visualizar, escrever e enviar dados, através da comunicação *Serial* do Arduino.

Utilizando estes blocos, é possível fazer uma ligação completa para comandar o carrinho por meio de qualquer dispositivo que utilize sistema operacional Android(pois este módulo, em específico, não é compatível com sistemas iOS).

Então, seguindo a programação, primeiramente criamos nossa variável "bluetooth_caractere", e em um segundo momento nos comunicamos com a porta serial, como mostra a Figura 33.

Figura 33 – Digrama de blocos para receber e armazenar dados do módulo Bluetooth



Fonte: Autor.

O primeiro bloco configura a comunicação *Serial* do Arduino, para que o seja possivel a comunicação entre dispositivos.

Em seguida, o bloco “SE” verifica se há dados disponíveis na porta serial. Caso tenha, o dado é armazenado na variável “bluetooth_texto”

A partir do momento que inserimos as variáveis e os dados do bluetooth em nosso programa, devemos configurar o aplicativo para utilizar os comandos desejados, de acordo com os caracteres (ou números) inseridos no programa.

Na Figura 34 é possível ver que, quando o sinal do bluetooth for igual a “F”, o carrinho irá andar para frente, e, por meio do aplicativo, será enviado essa letra “F” ao carrinho. Cabe lembrar que é necessário configurar as setinhas do aplicativo na função controle, com o caractere “F” para o carrinho andar para frente.

Figura 34 – Diagrama de blocos do recebimento para comparação do sinal recebido com o movimento desejado

Fonte: Autor.

Pode-se verificar na Figura 34 que da mesma forma que o caracter “F” faz com que o carrinho se mova para a frente, o carácter “R” fará com que o carrinho se mova para trás, o carácter “D” fará com que o carrinho se mova para a direita, o caracter “E” fará com que o carrinho se mova para a esquerda, e o caracter “P” fará com que o carrinho fique parado. Neste vídeo mostramos o carrinho sendo controlado por Bluetooth, nele será apresentado o código para comando por Bluetooth e o funcionamento do carrinho.

CARRINHO GUIADO POR RADAR 4

Neste capítulo vamos estudar como utilizar o sensor ultrassônico de distância e os servo motores (citados também na seção 1.1) presentes no Kit Escola Maker, dando autonomia ao carrinho robô para que o mesmo ande para frente e, caso chegue perto de algum obstáculo, ele pare, olhe para os lados e decida um caminho para ir. Para este projeto utiliza-se o sensor de distância para detectar os objetos, e o servo motor para fazer com que o sensor gire e olhe para os lados, para calcular a área com mais espaço para desviar dos obstáculos.

4.1 PROGRAMANDO RADAR

Assim como na programação do bluetooth, usaremos os blocos de variáveis, e os blocos de controle "SE, FAÇA", porém neste caso do sensor devemos criar duas funções a mais no nosso código, a função para ler o radar, e a função para escolher o lado que o carrinho deve seguir.

Função para ler o Radar: A função para ler o radar, é importante para que possamos acionar o sensor de distância, e dentro da função já determinamos o pino de

acionamento e o pino de eco, como mostrado na Figura 35:

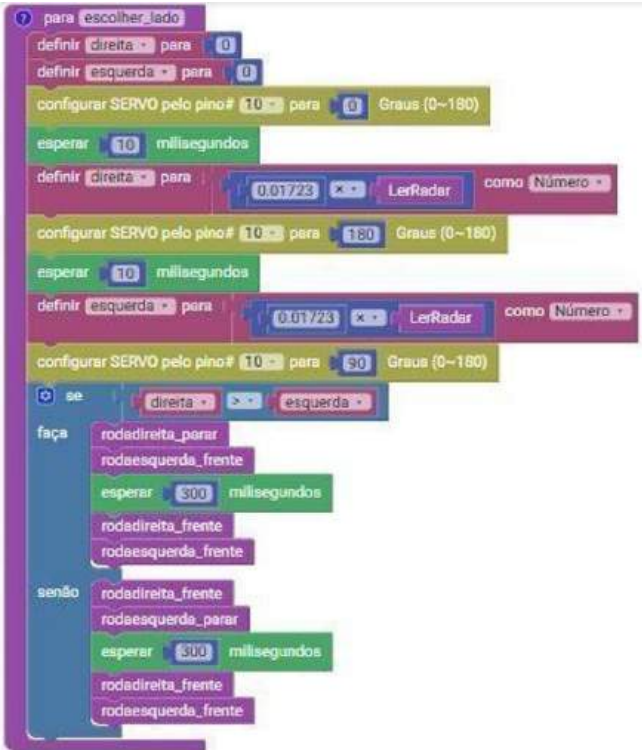
Figura 35 – Função para ler o Sensor de distância

Fonte: Autor.

Dentro desta função foi determinado um pulso de 100 microssegundos para o pino 12, o acionador, e retorna este pulso como sinal "ALTO", para o eco.

Função para escolher o lado: Como o próprio título já diz, utilizamos esta função para escolher o lado que o carrinho anda. Na função é incluído o acionamento do servo motor e o ângulo de giro, as variáveis "esquerda" e "direita" e dentro delas também é utilizada a função criada anteriormente "LerRadar", para armazenar a distância dentro das variáveis, como é mostrado na Figura 36:

Figura 36 – Função para escolher o lado



Fonte: Autor.

Primeiramente devemos configurar o pino do servo como o "10", como visto na Figura 35, e ele gire 0 graus (ele irá virar o sensor para direita). Então armazenamos o valor lido no radar em centímetros, bastando, para isso, multiplicar o sinal do radar por 0.01723 (número extraído através do tempo do sinal multiplicado pela velocidade do som, dividido por 2).

Depois gira-se o servo para o outro lado e repete a mesma lógica, porém desta vez armazenamos os valores calculados na variável "esquerda".

Por fim, utilizou-se um bloco "Se faça, senão", onde se a distância da direita for maior do que a da esquerda, o carrinho vira para a direita, se não for (no caso esquerda maior do que a direita), o carrinho gira para a esquerda.

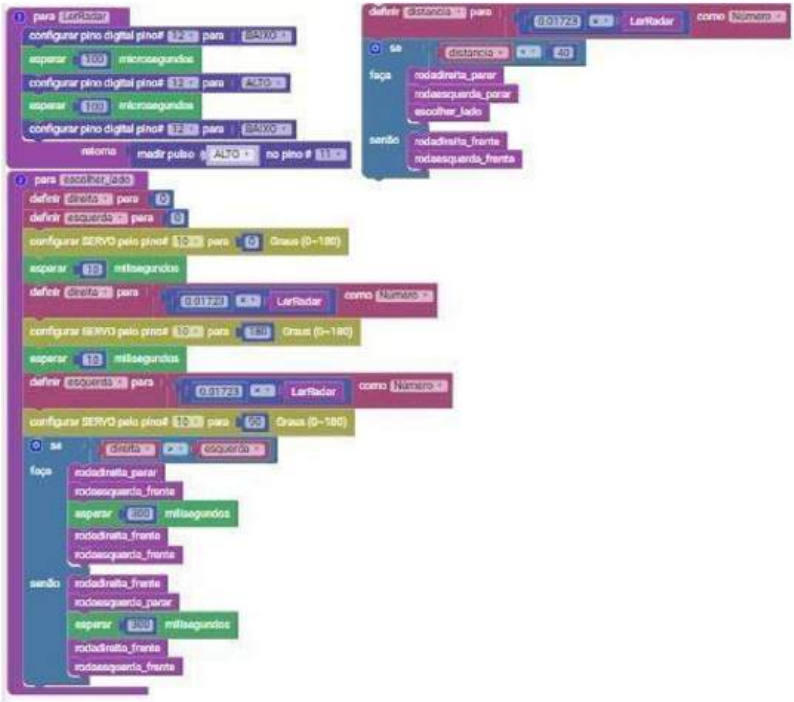
Depois de criar as funções necessárias para o desejado, deve-se criar o laço de repetição (*void loop*), como mostrado na Figura 37, para dar vida ao carrinho, neste *loop* será incluída uma nova variável para determinar a distância mínima, onde o robô irá parar e escolher o lado.

Figura 37 – Loop principal

Fonte: Autor.

Como podemos ver, calcula-se esta distância em centímetros, e, em seguida, utiliza-se um bloco de "Se faça, senão". Então se a distância for menor que 40 centímetros, o carrinho irá parar e realizar a função escolher lado, senão ele irá continuar andando para a frente. A estrutura completa do algoritmo é mostrada na Figura 38.

Figura 38 – Montagem completa do Radar



Fonte: Autor.

5 CONCLUSÃO

O material apresentado visa auxiliar no aprendizado de docentes da rede municipal de ensino e seus discentes (assim como *hobbistas*) em seus primeiros passos na robótica. Esperamos que com este conteúdo seja possível auxiliar os leitores sobre as ferramentas apresentadas, a começar pela plataforma Arduino, os seus componentes e o ambiente de programação utilizado, e por fim, mas não menos importante, a montagem do carrinho robô contido nos kits de robótica distribuídos para as escolas municipais gaúchas.

REFERÊNCIAS

AMARIEI, Cornel. **Arduino Development Cookbook**. Birmingham, Reino Unido: Packt Publishing Ltd, 2015. E-book. Acesso em: 28 jun. 2022.

ARDUINO. **Arduino Documentation.** Itália. Disponível em: https://www.arduino.cc/reference/pt/. Acesso em: 26 jun. 2022.

ARDUINO. **Arduino Documentation - Arduino UNO R3**. Disponível em: https://docs.arduino.cc/hardware/uno-rev3. Acesso em: 5 jun. 2023.

ARDUINO - Código bluetooth e funcionamento do Robô (2021). 1 vídeo (5 min e 14 seg). Santa Maria, Equipe Nightwind, 2021. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/ol4bvfG8yt4. Acesso em: 29 jun. 2022.

ARDUINO - Componentes e suas funções (2021). 1 vídeo (9 min e 13 seg). Santa Maria, Equipe Nightwind, 2021. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/83aqfs6_qtk. Acesso em: 29 jun. 2022.

ARDUINO- Conexões elétricas do robô (2021). 1 vídeo (7 min e 44 seg). Santa Maria, Equipe Nightwind, 2021. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/svmuxxX8Ir0. Acesso em: 29 jun. 2022.

ARDUINO - Explicando o Microcontrolador Arduino (2021). 1 vídeo (5 min e 28 seg). Santa Maria, Equipe Nightwind, 2021. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/GrN2dG3-Tto. Acesso em: 29 jun. 2022.

ARDUINO - Funções roda direita (2021). 1 vídeo (3 min e 50 seg). Santa Maria,

Equipe Nightwind, 2021. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/RPQvUQW0OWg. Acesso em: 29 jun. 2022.

ARDUINO - Movimento completo das rodas (2021). 1 vídeo (3 min e 50 seg). Santa Maria, Equipe Nightwind, 2021. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/eJqhyebg1x0. Acesso em: 29 jun. 2022.

ARDUINO - Montagem mecânica do robô (2021). 1 vídeo (11 min e 49 seg). Santa Maria, Equipe Nightwind, 2021. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/l53w3XOnOhI. Acesso em: 29 jun. 2022.

ARDUINO - Passando o código do NightwindBlockly para o Arduino (2021). 1 vídeo (3 min e 57 seg). Santa Maria, Equipe Nightwind, 2021. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/yZIMhQ5_QF0. Acesso em: 29 jun. 2022.

ARDUINO FORUM. Disponível em: https://forum.arduino.cc/. Acesso em: 25 jun. 2022.

BLOCK.INOock.ino. **Plataforma para programação de Arduino**. Disponível em: https://play.google.com/store/apps/details?id=org.blockinoandroid. blockino. Acesso em: 25 jun. 2022.

INSTALAÇÃO do software NightwindBlockly no Windows (2022). 1 vídeo (2 min e 56 seg). Santa Maria, Equipe Nightwind, 2022. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/c3EeelNl0nE. Acesso em: 29 jun. 2022.

INSTALAÇÃO NightwindBlockly no Linux (2022). 1 vídeo (2 min e 41 seg). Santa Maria, Equipe Nightwind, 2022. Publicado pelo canal NIGHTWIND Criatividade e Inovação. Disponível em: https://youtu.be/joBxdxki2XQ. Acesso em: 29 jun. 2022.

SCRATCH. **Free programming language**. Disponível em: https://scratch.mit.edu/. Acesso em: 25 jun. 2022

Elemento gráfico abstrato, capa e miolo:
FREEPIK. **Environment instagram posts.** Disponível em: https://www.freepik.com/free-vector/environment-instagram-posts_10280215.htm. Acesso em: nov. 2022.

Ilustração capa e folha de rosto:
MACROVECTOR. **Computer Service Pattern.** Disponível em: https://www.freepik.com/free-vector/computer-service-pattern_9376831.htm. Acesso em: nov. 2022.

UFSM

PRE