Título: STOP THINK & GO - Integração em Rede Inteligente IOT (Internet das Coisas)

Autores: 1911322473 Danilo Faria Dutra; 1911321861 Leonardo Marques dos Santos.

Orientadores/ Banca Examinadora: Prof. Ms. Angela Saemi K. Takesaki

Área do Conhecimento: Linha de Pesquisa Sistema de Computação: Ciência da Computação, Automação e Robótica

Introdução: Em um mundo globalizado, em grande desenvolvimento, é normal que algumas metrópoles possam enfrentar constantes engarrafamentos ou tenham dificuldades no trânsito local, causando problemas rotineiros, como engarrafamentos, acidentes automobilísticos, envolvendo inclusive pedestres. Os acidentes de trânsito já receberam uma atenção especial da Organização das Nações Unidas (ONU), que **enfatizaram:** "Uma das metas dos Objetivos de Desenvolvimento Sustentável ODS é a redução, pela metade, do número global de mortes e lesões no trânsito, até 2020"- (OPAS/OMS, 2018). No Brasil aumentaram-se cerca de 16% os números de acidente no primeiro trimestre de 2020 comparado com o mesmo período do ano passado (Tribuna de Ituverava, 2020), e mesmo após o estabelecimento do início de quarentena, os acidentes de trânsito com motociclistas aumentaram e acabaram por se tornarem as principais vítimas de acidentes de trânsito em São Paulo (G1 Globo, 2020), ou seja, justificando que há uma necessidade, um olhar mais apurado para esse tipo de problema, pois englobam vários atores relacionados ao trânsito.

De acordo com a ABI Research, uma empresa de pesquisas, em 2025 pelo menos oito milhões de carros autônomos estarão circulando pelo planeta e logo após 5 anos, em 2030, a previsão de comercialização destes carros, completamente autônomos, gira em torno dos 15%, caso as questões tecnológicas e regulatórias evoluam a contento, prevê a consultoria McKinsey. São muitos os benefícios que a automação de carros e integração em redes inteligentes IOT (*Internet Of Things*) podem gerar, como por exemplo redução no gasto de tempo, de congestionamentos por conta de um uso mais racional e planejado da infraestrutura urbana, entretanto nenhuma é mais importante que a possibilidade de redução no número de acidentes causados por falhas humanas, que segundo a OMS (Organização Mundial da Saúde), equivale a 90% dos acidentes automobilísticos.

O projeto propõe uma tecnologia que possibilite a automatização do sistema de semáforos junto aos automóveis, de modo a agilizar os processos realizados em trânsito, evitando possíveis acidentes e dinamizando o trânsito.

Desenvolveremos um sistema que automatiza semáforos e automóveis, ambos integrados em uma rede que gerencia e automatiza seus comandos, a fim de agilizar e dinamizar o modo em que o trânsito é conduzido atualmente.

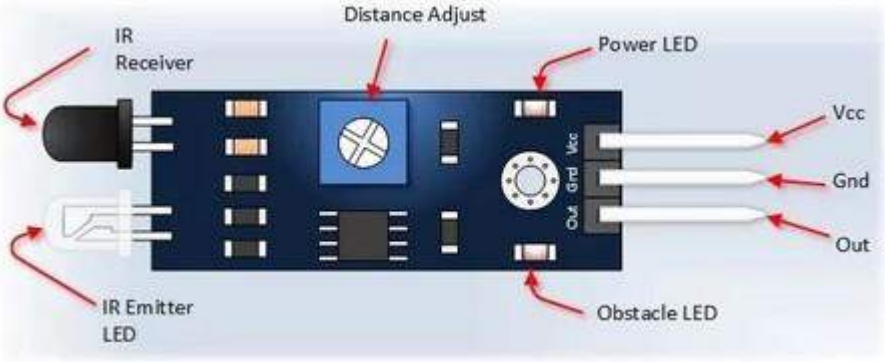
Objetivo: O objetivo do projeto é propor um sistema inteligente que gerencie e agilize o trânsito através de automóveis e semáforos automatizados, gerando benefícios em relação a tempo, custos, qualidade e segurança.

Metodologia: Com base na interpretação de dados de portais de notícia, que mostra o aumento de acidentes envolvendo automóveis, surgiu a ideia do projeto no qual irá automatizar o controle dos semáforos e as interações que os automóveis possuem entre si e o trânsito.

SENSOR INFRAVERMELHO REFLEXIVO DE OBSTÁCULO

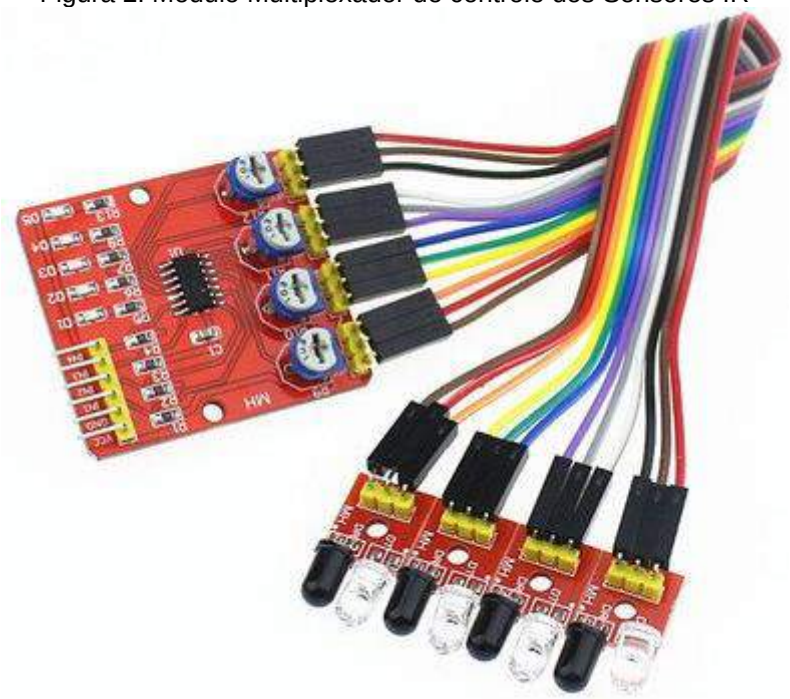
Os sensores infravermelho utilizam-se de uma faixa de luz não visível aos olhos humanos, portanto não conseguiremos identificar faixas de luzes emitidas pelo sensor. Na Figura 1 pode-se visualizar o módulo, no qual existe um Emissor LED, que envia as faixas de luzes em uma direção, a faixa é refletida pela barreira, que neste caso é o solo, e o Fotodiodo receptor identifica a faixa advinda do LED emissor infravermelho.

Figura 1: Módulo Sensor infravermelho com ajuste trimpot acoplado.



Fonte: https://arduinoeasy.wordpress.com/aplicacoes/sensor-infravermelho-de-obstaculo-com-lm393/

Figura 2: Módulo Multiplexador de controle dos Sensores IR

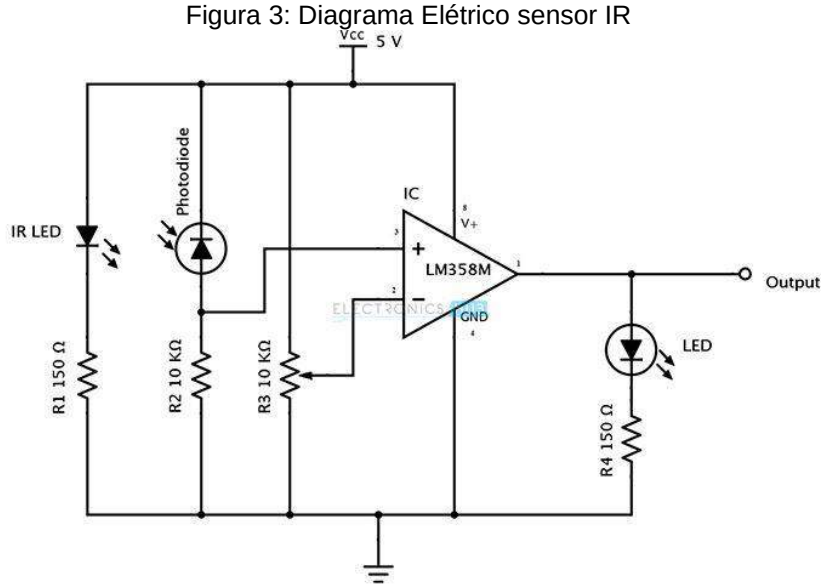


Fonte: https://www.eletrogate.com/modulo-seguidor-de-linha-com-4-sensores-infravermelho

FUNCIONAMENTO

Um dos principais componentes eletrônicos da PCI dos sensores IR é o CI LM393, muito utilizado em outros sensores, tais como sensor de gás, sensor fotoresistor, sensor óptico e sensor de velocidade, pois os mesmos utilizam de sua função comparadora de sinal.
O sensor através do LM393 faz o uso do comparador de sinal, para obter a leitura e identificar o valor de Entrada e Saída (I/O), tendo seu valor de limite regulado por um trimpot acoplado, mas que no caso do projeto, foi utilizado um Módulo Multiplexador, e no mesmo há os trimpots de regulagem, como mostrado na imagem acima.

Figura 3: Diagrama Elétrico sensor IR



Fonte: https://www.electronicshub.org/ir-sensor/

A distância de atuação dos sensores costumam variar entre 5 a 30 milímetros, e a quantidade de luz infravermelha recebida varia de acordo com a cor, material, forma e posição do obstáculo, não havendo assim possibilidade de medição de distância do obstáculo.

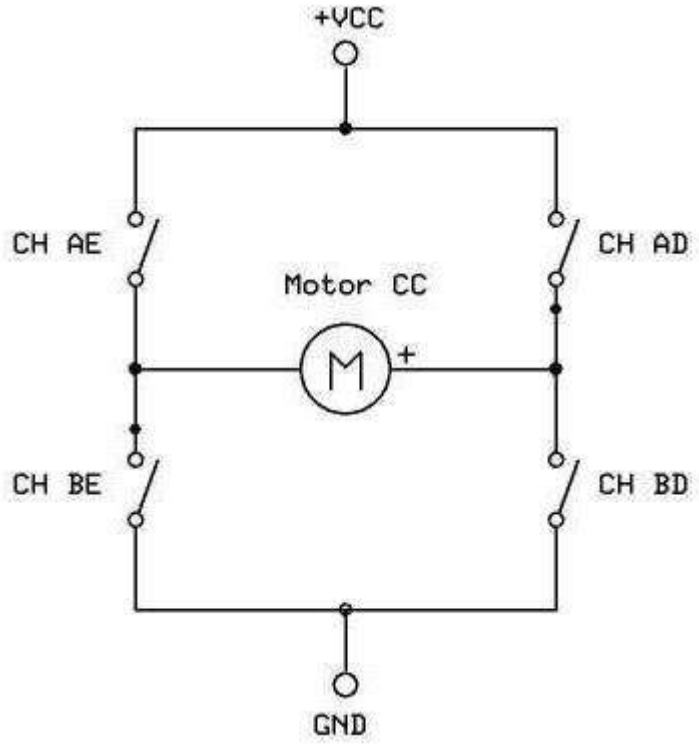
ESPECIFICAÇÕES TÉCNICAS

O sensor e o módulo em um todo funciona no nível lógico de 3.3v a 5v, podendo ser alimentado então pela saída de 5v do Arduino (caso haja uma alimentação externa, é necessário o jumpeamento do GND ao GND do microcontrolador para equiparar o nível lógico).
A entrada digital pode ser ligada em qualquer I/O digital do microcontrolador. O projeto utilizou-se das portas 8, 7, 5 e 4, que com exceção da porta 5, todas não são PWM, deixando as I/O PWM reservadas para o controlador da ponte H.

PONTE H DUPLA L298N versão mini

Módulos Ponte H ou ponte H construída são de extrema utilidade quando se trata de motores CC ou de passos, pois com ela conseguimos controlar a direção de rotação do motor e sua velocidade em utilizando PWM.

Figura 4: Diagrama elétrico de uma ponte H.



Fonte: https://blog.eletrogate.com/guia-definitivo-de-uso-da-ponte-h-l298n

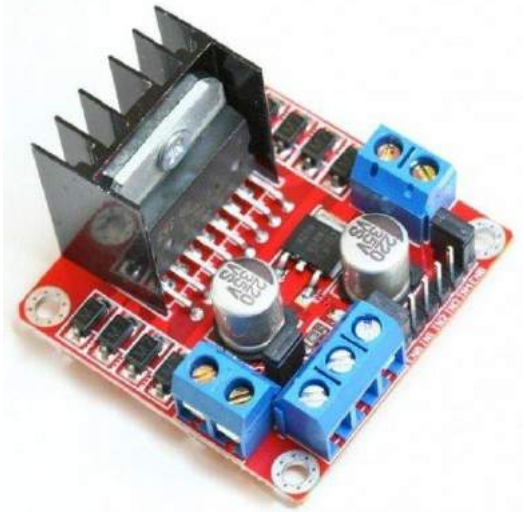
FUNCIONAMENTO

Para entendermos como funciona uma Ponte H, precisamos ter em mente como funciona um motor CC.
Em corrente contínua (DC), a corrente e tensão correm em um único sentido, portanto o Motor recebe a corrente do pólo positivo, passando por todo o motor e suas bobinas internas, e terminam o ciclo gerando tensão 0 ou GND e repetem o ciclo novamente até que a energia seja interrompida, fazendo com que o motor gire em um único sentido, podendo variar de acordo com a posição dos pólos. Por exemplo, se ligarmos o polo positivo (+) ao terminal 1 e o polo negativo (-) ao terminal 2 do motor, iremos supor que o motor gire em sentido horário, porém se invertermos os polos, ligando polo positivo (+) ao terminal 2 e polo negativo (-) ao terminal 1, o motor irá girar em sentido anti-horário, pois invertemos o sentido da corrente. Neste ponto é que a Ponte H é de extrema utilidade, pois conseguimos alternar o sentido da corrente com seu circuito projetado para tal finalidade.
Uma ponte H comumente pode ser criada a partir de transistores, porém para o projeto utilizamos um Módulo com o CI L298N para facilitar o uso embarcado e prototipagem do projeto.

O Módulo Ponte H L298N pode ser encontrado em duas versões, uma mais comum que contém um dissipador de calor acoplado ao CI L298N e a versão mini, que é a utilizada neste projeto.

Figura 5: LN298N versão Mini

Figura 6: LN298N versão com dissipador de calor



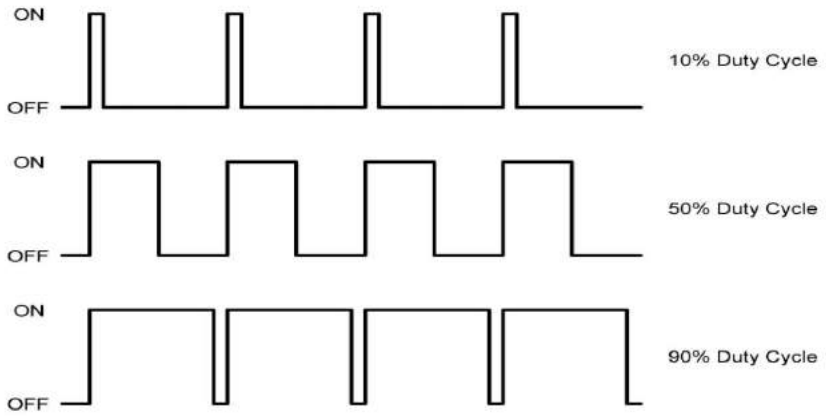
Fonte: https://www.eletrogate.com/ponte-h-dupla-l298n e https://www.eletrogate.com/mini-ponte-h-dupla-l298n

O motivo de utilizarmos a versão mini para este projeto se dá pela necessidade prática de uso, pois os motores com caixa de redução trabalham com tensão de 3v a 6v e necessitam de $\leq$ 200mA, podendo oscilar $\pm$ 10% RPM. O módulo mini aceita um nível lógico de 2v a 10v de bateria externa e a saída de cada canal é de 1,5A, sendo que a versão mais comum trabalha com 4v até 35v de bateria externa e 2A de saída em cada canal, não havendo necessidade, pois haveria desperdício de energia.

PWM

A Modulação por Largura de Pulso ou PWM (Pulse Width Modulation) é utilizada para controlar a entrega de tensão ao equipamento conduzido, ou seja, utiliza-se do Duty Cicle para controlar a onda quadrada o pulso.

Figura 7: Modulação PWM em Duty Cicle



Fonte: http://www.mecaweb.com.br/eletronica/content/e_pwm#:~:text=PWM%20significa%20%22Pulse%20Width%20Modulation,controle%20de%20pot%C3%AAncia%20ou%20velocidade.

Na primeira modulação da figura, vemos uma onda com um ciclo em 10%, na segunda vemos um ciclo de 50% e por fim na terceira vemos um ciclo de 90%.

O cálculo é feito pela seguinte fórmula matemática:

$$\textit{Duty Cycle} = 100\ x\ \frac{Largura\ do\ Pulso}{Período}$$

***Duty-Cycle*:** Valor em (%)
Largura do pulso: Tempo em que o sinal está ligado.
Período: Tempo de um ciclo da onda

Vamos supor que o motor trabalhe em uma frequência de 6 kHZ, ou 6000 HZ e tensão de 6v. Portanto 16,7 microsegundos . O motor ficou desligado em 5 microsegundos, portanto ficou ligado por 11,7 microsegundos.

$$\textit{Duty Cycle} = 100\ x\ \frac{11,7}{16,7} = 70\%$$

Portanto descobrimos que o *Duty Cycle* é de 70%, e aplicamos ao cálculo de Tensão Média:

Tensão Média = Tensão Aplicada x *Duty Cycle*

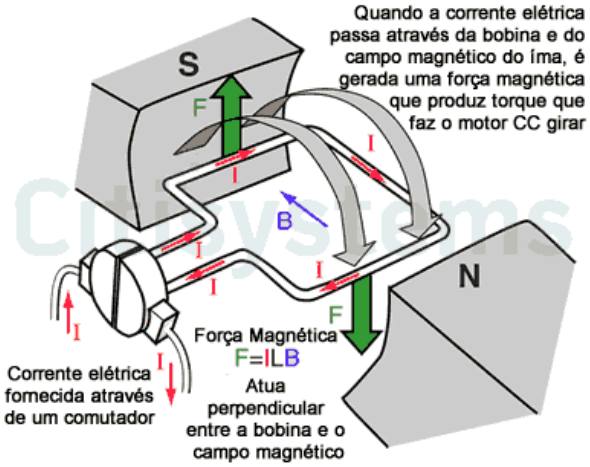
Neste caso, **Tensão Média** = 6v x 0,7
Tensão Média = 4,2v

Portanto a tensão média enviada ao motor será de 4,2v neste exemplo, fazendo com que o motor não opere em sua tensão máxima, controlando assim sua velocidade de giro através do PWM.

MOTOR CC

O funcionamento de um motor CC (Corrente Contínua) já foi abordado brevemente na explicação sobre a Ponte H, em que basicamente a corrente é percorrida pelas bobinas do motor em um sentido único conforme Figura 1.. A alimentação do motor pode ser proveniente de uma bateria ou qualquer outro tipo de alimentação CC.

Figura 8: Funcionamento de um motor CC



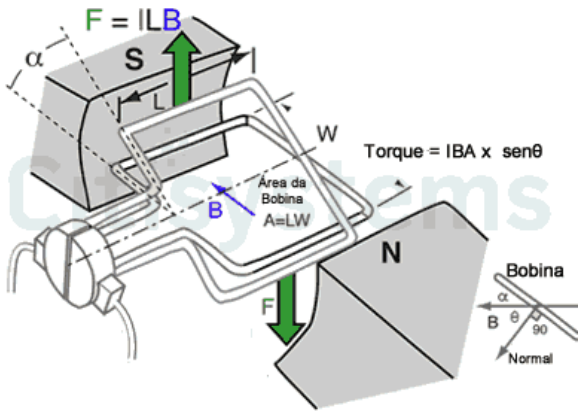
Fonte: https://www.citisystems.com.br/motor-cc

FUNCIONAMENTO

Na figura acima é mostrado a constituição dos motores, havendo um estator que é composto por dois imãs, um Norte e outro Sul e o rotor é representado por uma bobina que é alimentada pelo comutador em que circula uma corrente I.

Quando há uma alimentação CC, é gerado uma corrente contínua que é passada para a bobina através do contato das escovas do comutador a essa bobina. O comutador é o instrumento que faz a ligação entre a fonte de alimentação de energia e o rotor do motor CC, nele há escovas condutoras que fazem o contato com o eixo girante do motor.

Figura 9: A força e torque do motor



Fonte: https://www.citisystems.com.br/motor-cc

Três fatores são essenciais para se levar em conta ao selecionar um motor para um projeto, e tal qual para este foi levado em conta: a velocidade, o torque e a tensão.

Velocidade do eixo

A tensão é aplicada para iniciar a rotação do eixo, que é proporcional (ω). Geralmente as especificações de velocidade do eixo, condizem à velocidade sem carga alguma, que é a velocidade máxima que um motor pode atingir quando não há nenhum torque a ser considerado.
Quando falamos de motores, utilizaremos o termo RPM (Revolutions per minute) ou Rotações por minuto. As rotações também podem ser representadas por radianos por segundo (rad/s) e muitas aplicações numéricas, radianos se torna mais interessante de se aplicar. A fórmula matemática é composta por:

$$\omega rad/s = \omega rpm \cdot (\frac{2\pi}{60})$$

Quando um motor não há perdas, a velocidade rotacional é proporcional à tensão fornecida, sendo:

$$\omega = j.V$$

onde j é uma constante de proporcionalidade, dada em rad/(s.V).

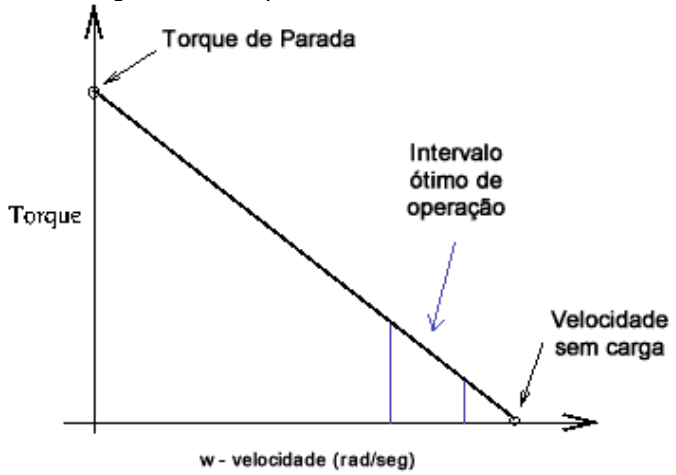
Torque de saída: A rotação do eixo gera uma força de rotação que é chamada de torque (τ).
Há dois tipos de torques: de partida e torque contínuo.
O torque de partida é o τ em que a velocidade do eixo é zero, significando que o motor não está em uma constante. Enquanto que o torque contínuo, o motor está em máximo τ em suas condições de normalidade. A fórmula é composta da seguinte maneira:

$$\tau = k \cdot I \quad ou \quad I = \tau / k$$

Sendo:

I = Corrente de indução
k = Constante de torque

Figura 10: Torque x Velocidade



Fonte: https://www.citisystems.com.br/motor-cc/

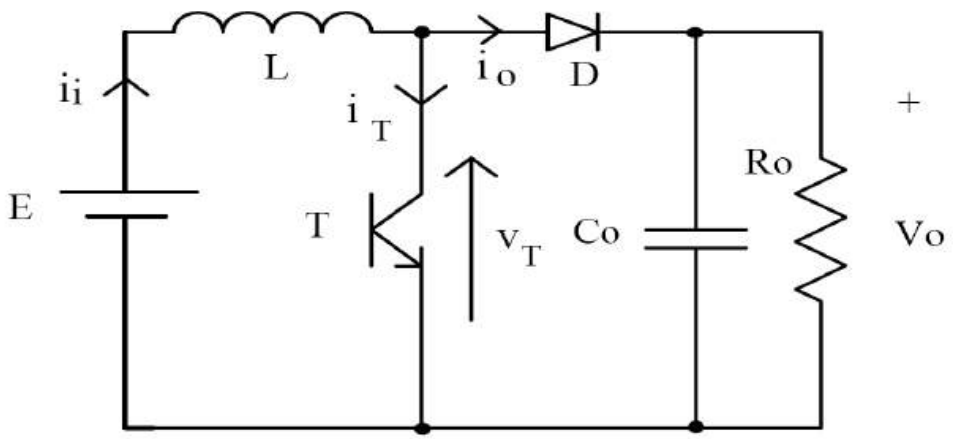
Tensão disponível: motores CC podem ser projetados para operar em uma específica tensão, havendo a necessidade de um limitador de tensão e/ou um circuito elétrico que reduza ao máximo o ripple de tensão, para que não haja falhas ou danos ao motor CC. É orientado sempre a verificação da faixa de tensão em que o motor pode trabalhar, para evitar problemas com o projeto.

Para aumento de torque do motor, foi acoplado uma caixa de redução a cada motor CC utilizado no projeto, no total 2 motores CC, um de cada lado (esquerdo e direito).

Step-up (Boost)

A função de um step-up ou boost, é elevar a tensão de entrada (Vi), gerando uma tensão maior na saída (Vo), ficando conhecido pelo termo elevador de tensão. Seu circuito é composto por no mínimo, um transistor (MOSFETs são muito utilizados), um diodo e um elemento de armazenamento de energia. São adicionados capacitores e indutores para ganho de desempenho.

Figura 11: Diagrama Elétrico de um *Step-up* (*Boost*)

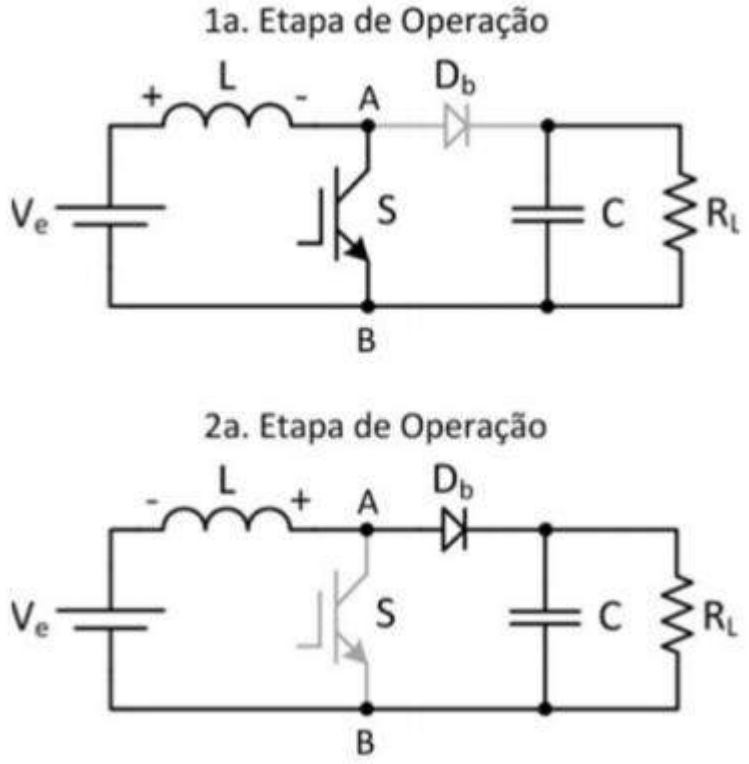


Fonte: http://www.tcc.sc.usp.br/tce/disponiveis/18/180500/tce-13052010-154507/publico/Rodrigues_Leandro_Gaspari.pdf

Neste caso, o boost não utiliza transformador, e sua saída possui a mesma polaridade que sua entrada.

De maneira parecida ao *Buck*, o conversor boost trabalha de forma contínua e descontínua. Em operação de modo contínuo, trabalha com duas etapas. Na primeira etapa, com chave fechada, o indutor L carregará e assumirá a tensão nominal do valor da fonte. Quando aberta a chave, o indutor estará com uma carga que produzirá uma tensão que se somará à tensão oferecida pela fonte para o carregamento do capacitor com uma tensão mais alta que a oferecida somente pela fonte inicialmente. Ao retorno da primeira etapa, o diodo será polarizado reversamente, fazendo com que o capacitor descarregue a tensão.

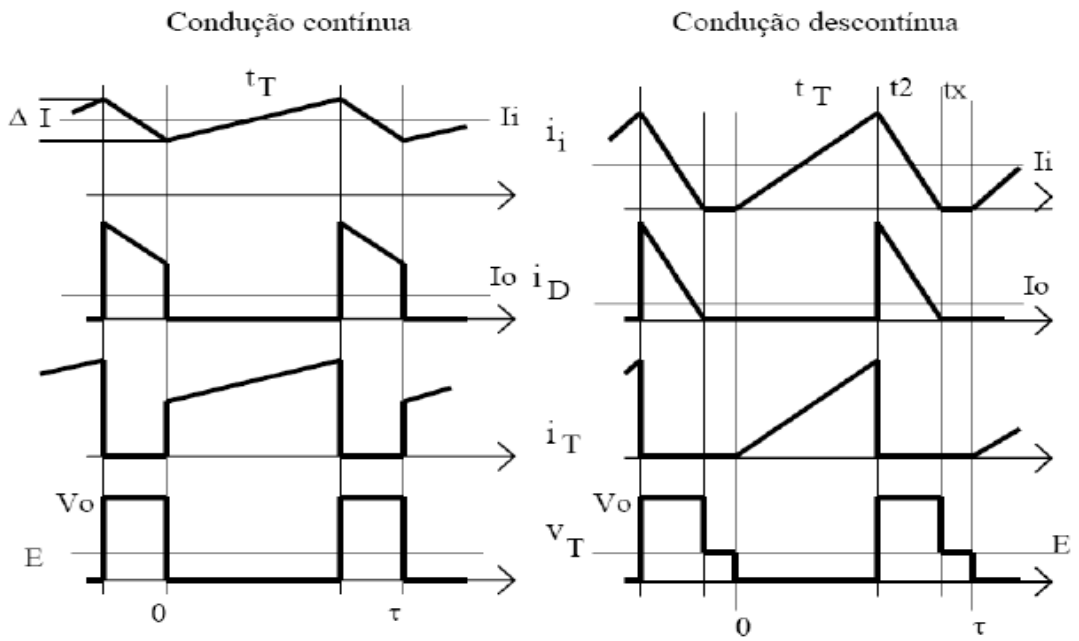
Figura 12: Onda quadrada e triângulo gerada pelo *step-up*



Fonte: https://bibliodigital.unijui.edu.br:8443/xmlui/bitstream/handle/123456789/4119/Felipe%20Barriquello%20Rosanelli.pdf?sequence=1

A figura abaixo demonstra as formas de onda do conversor, condução contínua e descontínua.

Figura 13: Onda quadrada e triângulo gerada pelo *step-up*



Fonte: http://www.tcc.sc.usp.br/tce/disponiveis/18/180500/tce-13052010-154507/publico/Rodrigues_Leandro_Gaspari.pdf

A equação para calcular a corrente nominal do conversor: $In = \frac{Pn}{Vs}$

Onde: **In** = Corrente nominal do conversor
Pn = Potência nominal
Vs = Tensão de saída

A corrente necessária para alimentarmos cada motor do projeto é de ≤200mA a 6v, portanto multiplicaremos por 2, pois existem dois motores, sendo um do lado esquerdo e outro do direito.
No modelo que utilizamos, MT3608, existe uma regulagem automática de tensão de saída, quando a tensão de entrada é alterada. Sua tensão de entrada pode variar de 2v~24v, com tensão de saída variando de 2,5v~28v, gerando uma eficiência de 91%. Sua corrente máxima é de até 2A.

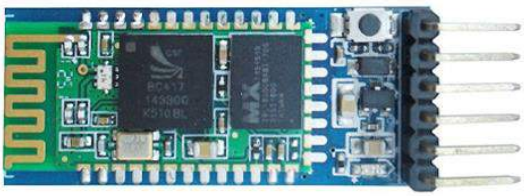
Figura 14: Step-up MT3608

Fonte: Autores, 2020

HC-05 (Módulo Bluetooth)

Muito utilizado em comunicação via rede 2.4GHz, os módulos *bluetooth* possuem diversos modelos e cada um com características e formas de trabalhos específicas e também parecidas dentre si. Por exemplo, o módulo HC-06 funciona de forma semelhante ao módulo HC-05, porém foi preterido pelo fato de trabalhar somente em modo *SLAVE,* uma vez que o HC-05 trabalha em modo *SLAVE, MASTER.*

Figura 15: Módulo HC-05



Fonte: Imagem de Internet, 2020.

Se diz que os módulos trabalham em 2.4GHz nominalmente, porém podem variar entre 2.4GHz a 2.485GHz.
O protocolo e especificações técnicas da tecnologia é determinado por um grupo de empresas e centros de desenvolvimento chamado *Bluetooth Special Interest Group,* e são responsáveis por lançarem versões atualizadas e novas funcionalidades. Portanto, é um protocolo licenciado e para que um produto possa estampar seu emblema, precisa estar de acordo com todas as especificações do protocolo.
Anteriormente já citados, os principais módulos de uso para gerar uma interface de comunicação entre o protocolo e o Arduino são os módulos HC-05 e HC-06.

As principais versões do protocolo são:

Tabela 1: Principais versões Bluetooth

Versão	Velocidade
1.2	1Mbit/s
2.0+EDR	3Mbit/s
3.0	25Mbit/s
4.0 (*Bluetooth Low Energy*)	25Mbit/s
5.0	50Mbit/s

Fonte: https://pt.wikipedia.org/wiki/Bluetooth

CONFIGURAÇÃO AT

Para o correto funcionamento dos módulos, é necessário a configuração correta de comunicação entre os módulos.
É necessário a utilização da IDE do Arduino, que pode ser acessada em http://www.arduino.cc.
Posteriormente a instalação da IDE, precisamos conectar o Arduino, de forma que a comunicação Serial possa ser feita. A configuração do módulo é feita através do monitor Serial e é necessário inicializá-lo em modo AT(Modo de configuração interna). Há também uma outra mudança necessária, que está na mudança de retorno para “Ambos NL & CR”.
A inicialização em modo AT é feita apertando e segurando o botão de reset externo antes de ter feito a ligação do circuito elétrico, fazendo com que o LED externo pisque em intervalos mais longos.

Há uma lista consideravelmente grande de comandos AT, porém para tal uso, o artigo irá se conter somente aos necessários para configurá-lo para tal modo.

Já na janela do monitor Serial, devemos verificar se a comunicação está sendo feita de forma correta, e para tal, vamos usar o comando **AT** e enviar. Caso esteja tudo certo, o monitor retornará um “OK”.
Posteriormente, os comandos abaixos devem ser enviados:

AT+UART=9600, 0, 0

Configura o *BAUD RATE* do módulo a trabalhar na frequência de 9600.

AT+CMODE=1

O comando AT+CMODE indica o tipo de conexão, sendo 1 aberta,aceitando qualquer dispositivo com a mesma faixa de comunicação, e 0 em comunicação fechada, aceitando somente dispositivos específicos.

AT+ROLE=0

Configura o modo de operação, sendo 0 em modo *SLAVE* e 1 em modo *MASTER*

AT+ADD= ENDEREÇO_MAC_ADDRESS

Configura o endereço *MAC ADDRESS* a enviar/receber dados, que no caso do projeto, a comunicação foi feita em modo *Simplex*, em que um dispositivo foi configurado como *MASTER* e outro em *SLAVE.*

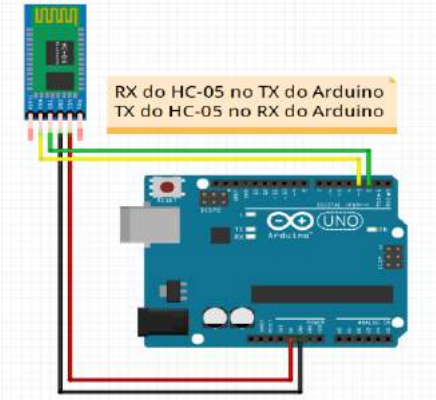
AT+RESET

Reinicia o módulo e o coloca em modo de conexão.

Posteriormente a configuração AT, os módulos deverão comunicar entre si automaticamente, uma vez que ligados.

Uma vez que os módulos estejam configurados, devemos também utilizar a biblioteca “SoftwareSerial.h” para podermos definir a porta digital 2 como RX e a porta digital 3 como TX, assim deixamos as portas de comunicação RX/TX livres para fazer o *upload* do código e testar possíveis erros e corrigi-los com mais eficiência.

Figura 16: Ligação eletrônica demonstrada no Fritzing, de acordo com o SoftwareSerial.h



Fonte: Autores, 2020.

Arduino UNO v.R3

O arduino foi desenvolvido em plataforma para prototipagem eletrônica. Possui um *hardware open source*, podendo ser distribuído e modificado livremente.

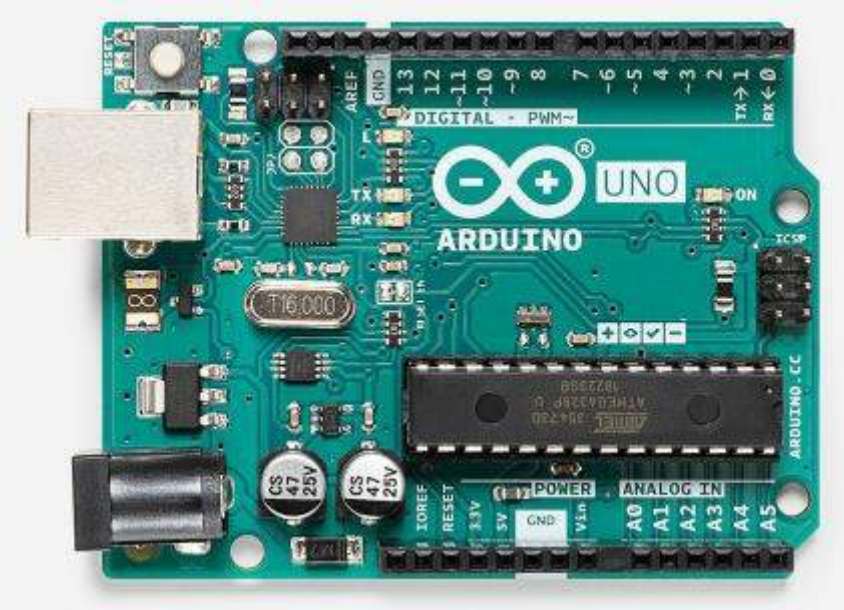
Projetada a partir de microcontroladores Atmel AVR, possui suporte de I/O (Entrada/Saída), e sua principal linguagem de programação é baseada em C/C++.
Um detalhe a ser considerado é seu *framework*, com a utilização da biblioteca "Arduino.h" incorporada a IDE e algumas funções principais já nativas na própria IDE, não sendo necessário o aprofundamento em baixo nível da linguagem ao microcontrolador.

Ao longo de sua existência a versão mais conhecida, o UNO, foi passando por transformações físicas, tanto em seu componentes, passando do encapsulamento DIP, para SMD, quanto na forma de I/O de comunicação, passando do RS232 para USB-B.

Figura 17: Arduino UNO *Serial*, com componentes em encapsulamento DIP e entrada RS232

Fonte:https://www.arduino.cc/en/Main/ArduinoBoardSerial

Figura 18: Arduino UNO rev3, versão mais atual.



Fonte: https://store.arduino.cc/usa/arduino-uno-rev3

Atmega328P

Existem dois encapsulamentos utilizados pelo modelo UNO: SMD e DIP.

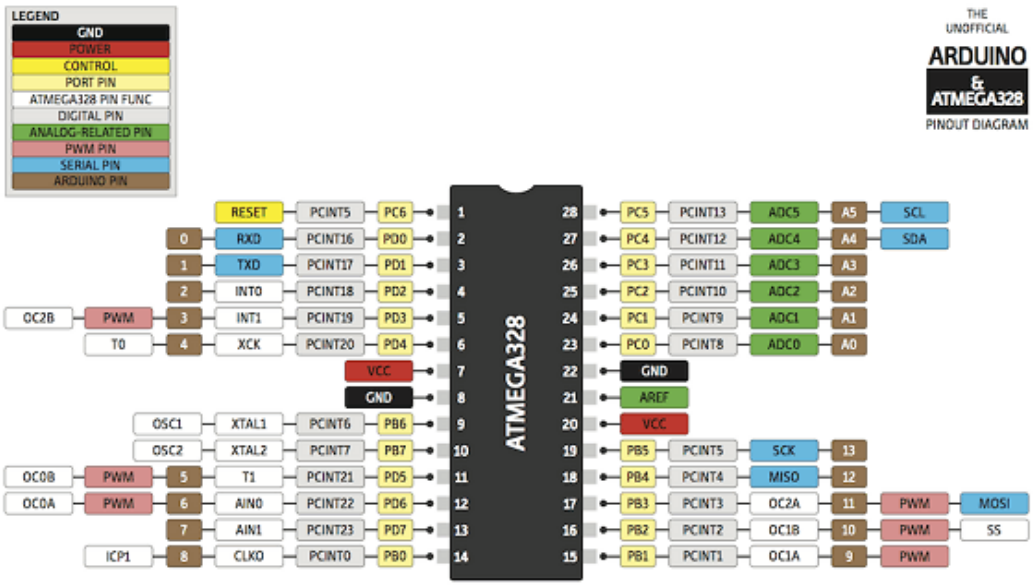
Figura 19: Na figura da esquerda temos um UNO com encapsulamento SMD e à direita um DIP

Fonte: https://boutique.semageek.com/en/68-arduino-uno-smd-r3.html

O microcontrolador ATMEGA328 trabalha em 8 bits, é um AVR, arquitetura RISC avançada. Conta também com 32KB de memória *Flash*, sendo que 512 bytes são utilizados pelo *bootloader*, 2KB de memória RAM e 1KB de memória EEPROM.
O microcontrolador pode operar em 20MHz, mas é mais comum trabalharem em 16 MHz em placas do UNO, que está conectado aos pinos 9 e 10 do microcontrolador.

Possui 28 pinos, sendo 23 destes em I/O.

Figura 20: “Pinagem” do ATMEGA328



Fonte: https://www.embarcados.com.br/arduino-uno

Tabela 2: Especificações Técnicas do ATMEGA328

Tensão de Operação	5V
Tensão de Entrada (Recomendada)	7-12V
Tensão de Entrada (Limite máximo)	6-20V
I/O Digitais	14 (6 PWM OUTPUT)
Entradas Analógicas	6
Corrente máxima de I/O	20 mA
Corrente máxima em 3.3V	50 mA
Memória Flash	32KB (500 bytes utilizados pelo bootloader)
SRAM	2KB
EEPROM	1KB
Clock	16MHz
LED interno	Pino 13

Fonte: https://store.arduino.cc/usa/arduino-uno-rev3

Portas I/O

De acordo com prototipagem do projeto, a tabela abaixo representa a utilização das I/O:

Tabela 3: Utilização das I/O do Arduino UNO no projeto

Item	Portas
Motor 1	6 (Sentido Horário)
	9 (Sentido Anti-horário)
Motor 2	10 (Sentido Horário)
	11 (Sentido Anti-horário)
Sensor Infravermelho 1	8
Sensor Infravermelho 2	5
Sensor Infravermelho 3	5
Sensor Infravermelho 4	4
RX	2
TX	3

Fonte: Autores, 2020

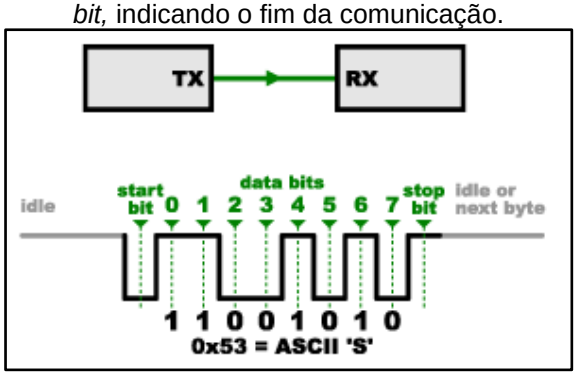
Hardware e Comunicação

O hardware foi testado e construído com base em comunicação UART, via rede 2.4GHZ, criando comunicação entre *MASTER* e *SLAVE*, utilizando microcontroladores *ATMEGA328P* e módulos de comunicação HC-05, podendo ser expandido para outros protocolos futuramente, como ESP-NOW, CAN, Rede *Mesh*, com a ampliação e utilização de ESP8266 e/ou ESP32.
A medida de verificação é feita em bits por segundos e é assimilada a velocidade de comunicação entre as partes. Caso uma comunicação assíncrona envie um *bit* em um *BAUD RATE* de 9600 bps, enviará um bit em 0,0001 segundos. A terminologia pode variar de acordo com o tipo de comunicação, por exemplo, em comunicações assíncronas é costumeiro chama-lá de *BAUD RATE,* e em síncronas, utiliza-se *clock*, pelo fato dos cristais osciladores estarem sincronizados entre si e gerarem o mesmo "*Clock*".
Como a comunicação assíncrona não se utiliza de sinais de *clock*, vindos dos cristais osciladores, a utilização de fios necessários para o sistema roda é bem

menor. Entretanto, como a comunicação não é sincronizada entre os *clocks* respectivos, a comunicação pode ser suscetível a erros e portanto parâmetros na sistematização são necessários para garantir uma boa comunicação. Uma configuração necessária é o ajuste do *BAUD RATE* na mesma frequência. É de extrema importância que os dispositivos trabalhem na mesma frequência. Caso necessário, é recomendável verificação em modo AT para garantir as mesmas configurações. Faz se necessário o uso do comando “AT+UART?”, que verifica a frequência do *BAUD RATE*, e a caso necessário, faz-se uso do AT+UART=FREQUÊNCIA, 0, 0, em que configura a frequência de trabalho do dispositivo. É necessário também o dispositivo estar em boot modo AT.

A comunicação via UART é feita da seguinte maneira: O TX envia um bit com valor 0 (chamado de s*tart bit*), em seguida envia os dados a serem transmitidos, o rx faz a leitura do pacote bit a bit, sendo de 5 a 9 bits de informações e 1 bit de paridade para evitar a recepção de erros, que neste caso é fechado com um sinal que pode varia de 0,5 a 2 *bits* (dependendo da configuração do dispositivo). Frisando sempre que a quantidade de dados a serem transmitidos (*bitrate*) varia de acordo com a oscilação de sinal (*BAUDRATE*).

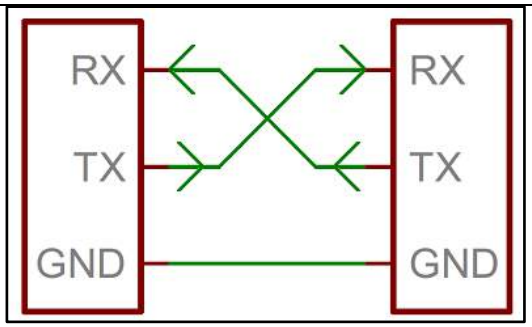
Figura 13: O TX envia o s*tart bit* indicando que há um ínicio de comunicação, envia os *data bits,* que representa os valores a serem enviados em forma de *bits* e ao término, envia o *stop bit,* indicando o fim da comunicação.



Fonte: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
Acesso em outubro 2020.

A comunicação serial envia e recebe toda a informação sequencialmente, o que permite haver um menor número de fios. Uma vez que a proposta foi de integrar sistemas em tecnologia *wireless*, e a comunicação foi feita no sentido *simplex*, não houve razão de uso do protocolo em paralelo.

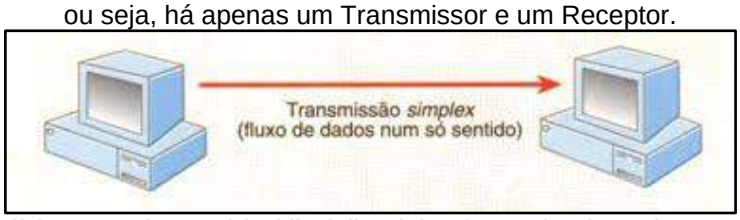
Figura 14: A comunicação RX é interligada a TX e a TX interligada a RX, GND interligado com GND, fazendo com que um lado comunicador envie dados assíncronos para para o outro lado receptor, que dependendo da configuração, pode ser *simplex*, *half duplex* ou *full duplex*.



Fonte: https://learn.sparkfun.com/tutorials/serial-communication
Acesso em outubro 2020.

O sentido de transmissão é ***Simplex***, ou seja, o TX (*Master*) envia os dados ao RX (*Slave*) e este reage de acordo com os dados recebidos do Transmissor como apresentado na Figura 3

Figura 15: Na comunicação simplex, somente um lado envia os dados a serem interpretados, ou seja, há apenas um Transmissor e um Receptor.

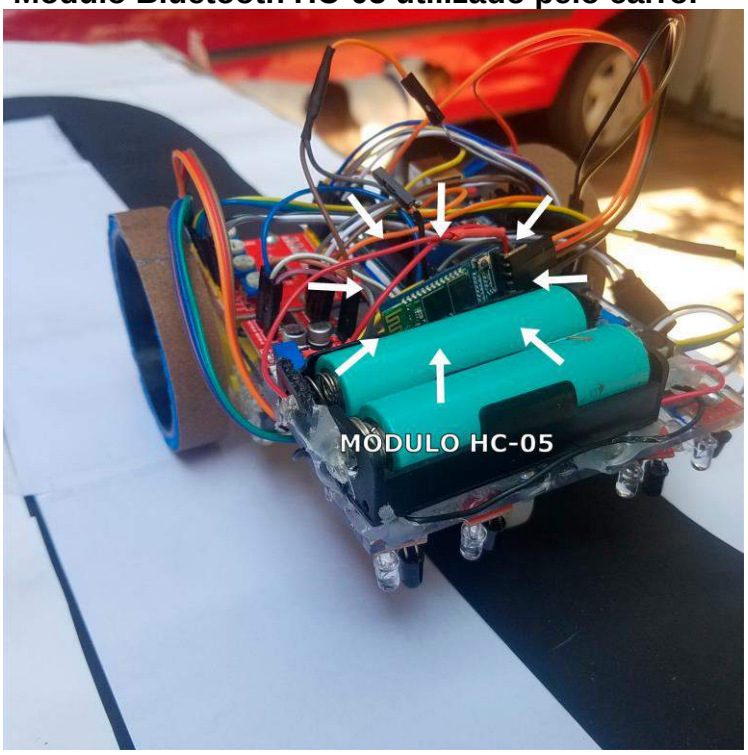


Fonte: https://sites.google.com/site/disciplinadeiccr/Home/9--tipos-de-transmissoes/2--transmisssoes-simplex-half-duplex-e-full-duplex
Acesso em outubro 2020.

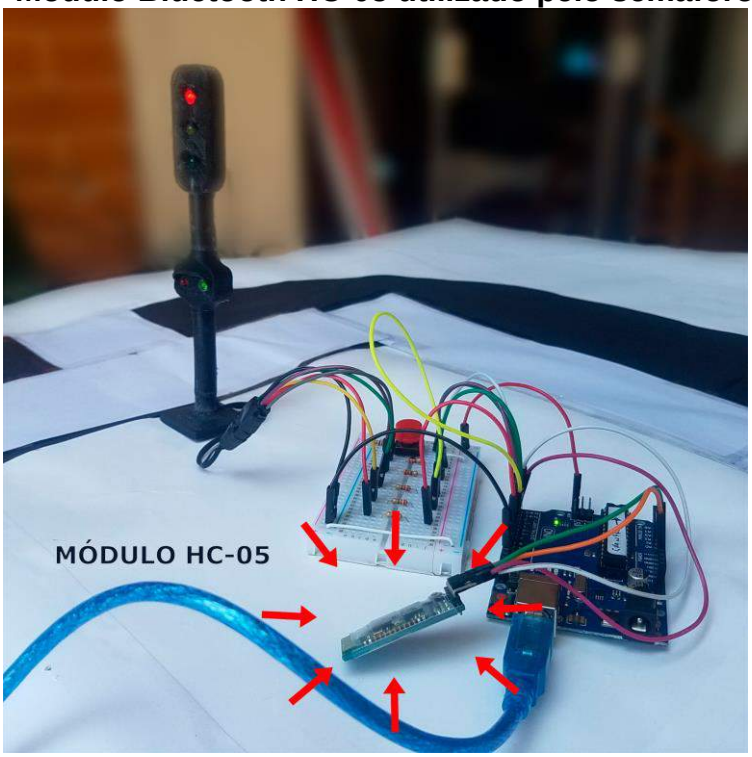
A estrutura do algoritmo foi pensada para que exista a possibilidade de haver comunicação de vários sistemas interligados em si.

Resultado e Discussão: O sistema distribuído responde corretamente ao que foi proposto, não houve falha na comunicação entre o Transmissor e o Receptor.

Módulo Bluetooth HC-05 utilizado pelo carro.



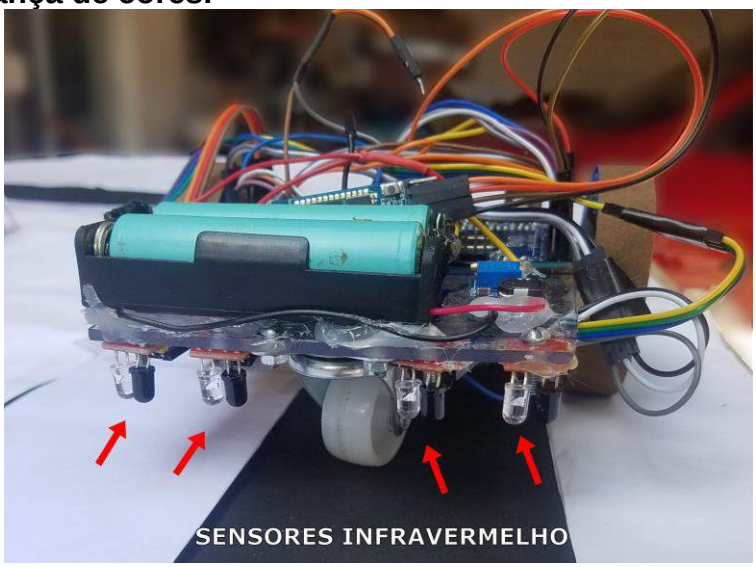
Módulo Bluetooth HC-05 utilizado pelo semáforo.



Em situações em que há falha no sinal enviado, por segurança o dispositivo RX desliga os motores e aguarda a resposta de sinal novamente. No caso de uma situação real, é planejado que o sistema entre em modo manual, até que haja uma nova conexão, não prejudicando assim o trânsito.

Sensores infravermelho que identificam presença ou

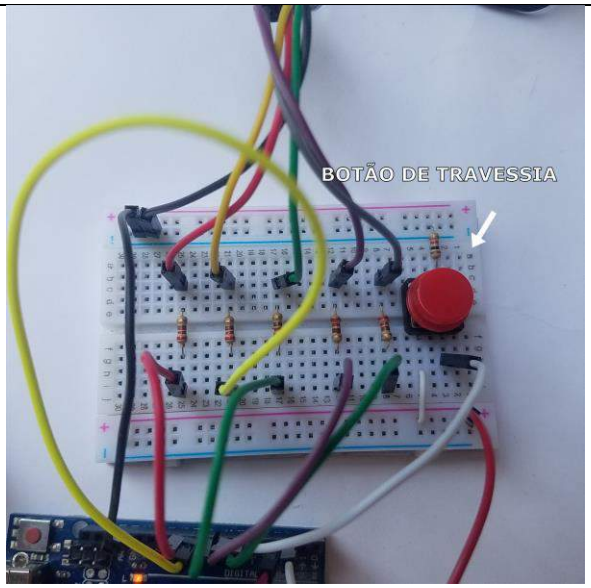
mudança de cores.



Os sensores funcionam de forma digital, ou seja, identificam ou não presença ou mudança de cor, não havendo assim sinais com valores contínuos, sendo valores discretos.

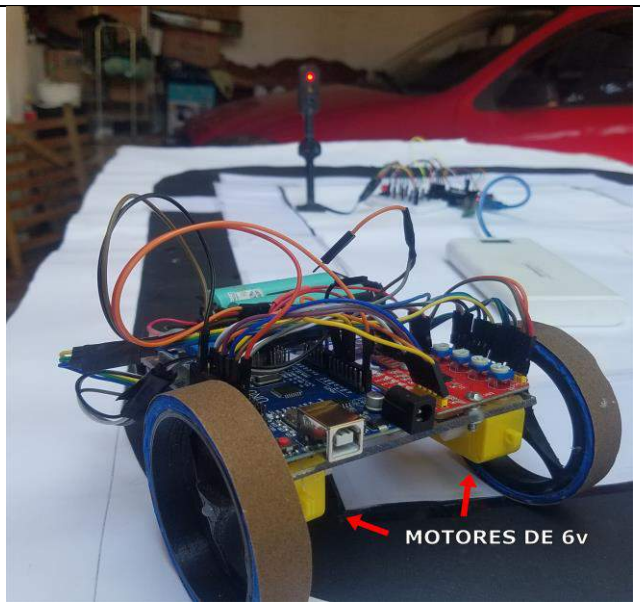
Sinalização do semáforo.

A sinalização responde corretamente aos sinais, não havendo atrasos (*delay*), uma vez que o microcontrolador executa duas funções “ao mesmo tempo”, não utilizando de funções “*delay*”.



Botão de travessia de pedestres.

Eletrônica do carro (vista de cima).



Motores 6v com caixa de redução (200 RPM máx em 6v).

Segue uma breve apresentação do Projeto:

STOP
THINK &
GO!

http://gg.gg/STOP_THINK_AND_GO

Conclusão: Concluímos que há uma grande brecha a ser explorada, se tratando de segurança pessoal e conjunta em trânsito brasileiro, e que há muita inovação em tecnologias voltadas para melhoria de performance dos carros, porém pouco é explorado em relação a segurança física e manutenção de vidas. Com os testes, foi possível verificar que o protótipo responde bem em relação a integração de sistemas, podendo se expandir-se para um conjunto de automóveis, tal como é composto o trânsitonormalmente. Possíveis melhorias podem ser implementadas, aumentando ainda mais a segurança física e aperfeiçoando a comunicação de dados.

Referências (formatação ABNT):

1. OPAS/OMS Brasil, Metas da segurança no trânsito e objetivos do desenvolvimento sustentável. Disponível em <https://www.paho.org/bra/index.php?option=com_content&view=article&id=2117:metas-da-seguranca-no-transito-e-objetivos-do-desenvolvimento-sustentavel&Itemid=779>. Acesso em: 04 de setembro 2020.
2. G1 Globo, Motociclistas se tornam as principais vítimas do trânsito em São Paulo durante a quarentena. Disponível em <https://g1.globo.com/sp/sao-paulo/noticia/2020/08/20/motociclistas-se-tornam-as-principais-vitimas-do-transito-em-sao-paulo-durante-a-quarentena-diz-estudo.ghtml>. Acesso em: 04 de setembro de 2020.
3. Tribuna de Ituverava, Aumentam casos de morte no trânsito no Brasil no primeiro trimestre de 2020. Disponível em <http://www.tribunadeituverava.com.br/aumentam-casos-de-morte-no-transito-no-brasil-no-primeiro-trimestre-de-2020/#:~:text=De%20janeiro%20a%20mar%C3%A7o%20de,Estat%C3%ADstico%20da%20Seguradora%20L%C3%ADder%2DDpvat>. Acesso em: 04 de setembro de 2020.
4.Robocore, Comparação entre protocolos Serial. Disponível em <https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html>. Acesso em: 25 de setembro de 2020
5. Wikipédia, Protocolo UART/USART. Disponível em <https://pt.wikipedia.org/wiki/USART>. Acesso em: 25 de setembro de 2020.
6. Newton C. Braga, Como funcionam as UARTs. Disponível em <http://newtoncbraga.com.br/index.php/telecom-artigos/1709->. Acesso em: 25 de setembro de 2020
7. Estadão, Eles estão chegando, veículos autônomos. Disponível em <http://patrocinados.estadao.com.br/arteris/eles-estao-chegando>. Acesso em 25 de setembro de 2020.
8. Mundo Projetado, Sensor Infravermelho de obstáculo com LM393. Disponível em <http://mundoprojetado.com.br/sensor-de-obstaculo-infravermelho>. Acesso em 23 de outubro de 2020.
9. Blog Master Walker Shop, Como usar com Arduino – Sensor Infravermelho Reflexivo de Obstáculo. Disponível em <https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-sensor-infravermelho-reflexivo-de-obstaculo>. Acesso em 23 de outubro de 2020.
10. Blog Eletrogate, Guia Definitivo de uso da Ponte H L298N. Disponível em <https://blog.eletrogate.com/guia-definitivo-de-uso-da-ponte-h-l298n>. Acesso em 23 de outubro de 2020.

11. MecaWeb, PWM - Modulação Por Largura de Pulso. Disponível em <http://www.mecaweb.com.br/eletronica/content/e_pwm#:~:text=PWM%20significa%20%22Pulse%20Width%20Modulation,controle%20de%20pot%C3%AAncia%20ou%20velocidade>. Acesso em 23 de outubro de 2020.
12. Citysystems, O que é PWM e Para que Serve? Disponível em <https://www.citisystems.com.br/pwm>. Acesso em 23 de outubro de 2020.
13. Citysystems, Motor CC: Saiba como Funciona e de que forma Especificar. Disponível em <https://www.citisystems.com.br/motor-cc>. Acesso em 23 de outubro de 2020
14. Leandro Gaspari Rodrigues, Estudo e desenvolvimento de um conversor DC-DC de Topologia Buck para aplicação Aeroespacial
15. Wikipedia, Bluetooth. Disponível em <https://pt.wikipedia.org/wiki/Bluetooth>. Acesso em 27 de outubro de 2020.
16. FilipeFlop, Como usar o Arduino Bluetooth HC-05 em modo mestre. Disponível em <https://www.filipeflop.com/blog/tutorial-arduino-bluetooth-hc-05-mestre>. Acesso em 09 de novembro de 2020.
17. Curto Circuito, Configuração do modo Master-Slave do HC-05 com Arduino. Disponível em <https://www.curtocircuito.com.br/blog/arduino-e-hc-05:-configuracao-master-slave> . Acesso em 09 de novembro de 2020.
18. Arduino.cc (Site Oficial), Documentação da Biblioteca “SoftwareSerial.h”. Disponível em <https://www.arduino.cc/en/Reference/softwareSerial>. Acesso em 09 de novembro de 2020.
19. Arduino.cc (Site Oficial), Constructor da Biblioteca “SoftwareSerial.h”. Disponível em <https://www.arduino.cc/en/Reference/SoftwareSerialConstructor>. Acesso em 09 de novembro de 2020.