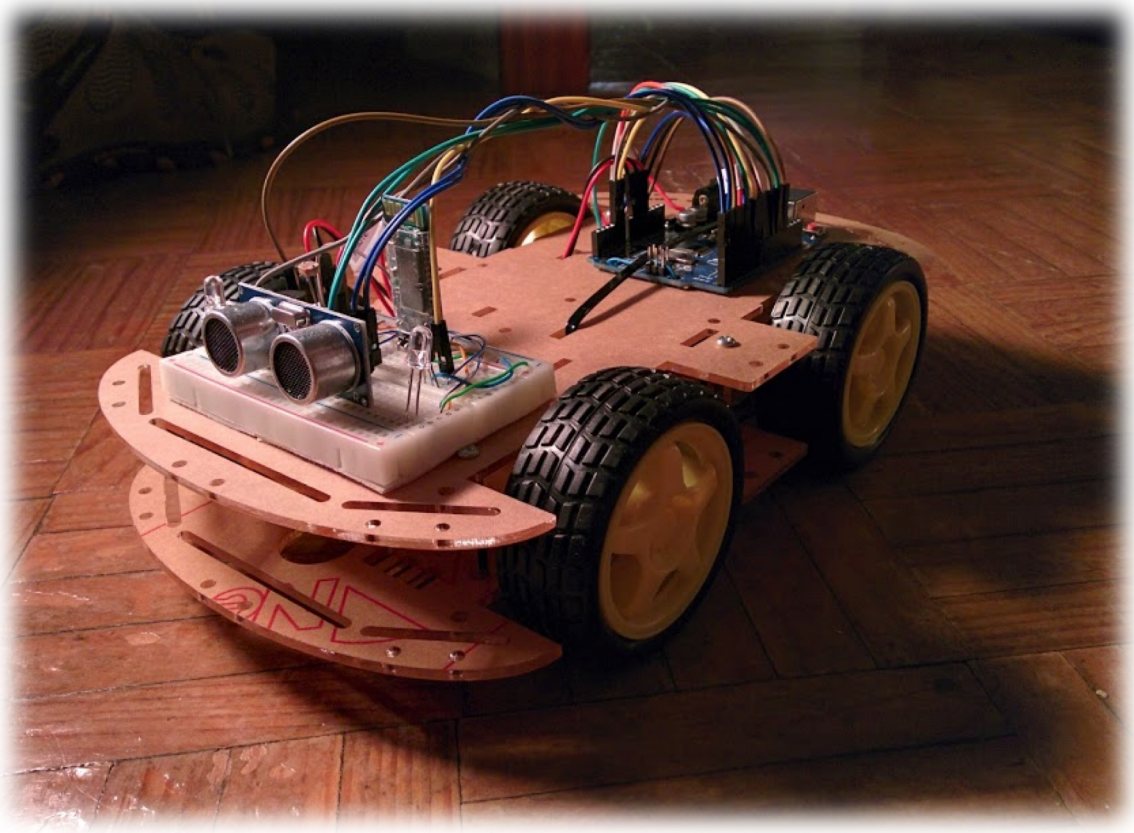
IPBeja
Instituto Politécnico de Beja

Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão



ARDUINO RC CAR

Simple Arduino RC Car

Autor: Miguel Rosa (6219)
Disciplina: Física Aplicada à Computação
Docente: Prof. Nuno Pereira
Ano Letivo: 2015/16

Beja, Janeiro 2016

Índice

Abstract

On this work, I'll create a RC Car with arduino platform. To control this car, I created an Android Application and connect both with Bluetooth. On this app, the user can control the movements (forward, backward, left, right and stop), turn on/off the LED's and play sounds from buzzer.

The main objetive was apply the knowledge acquired from class, some independente work/search on internet, books, etc.

Keywords: RC Car, Arduino, Components, Android

Introdução

No âmbito da disciplina de Física aplicada à computação foi-nos dada a possibilidade para a avaliação da componente teórica a realização de um trabalho prático que permitisse aplicar os conhecimentos adquiridos nas aulas ao longo do semestre.

De forma a colmatar as minhas lacunas nas componentes de eletrónica e eletricidade a minha proposta de trabalho é a criação de um carro telecomandado via Bluetooth (também designado por RC Car), onde irei acoplar sensores que permite a recolha de vária informação que será útil ao utilizador. Irei usar uma plataforma de prototipagem e uma aplicação para dispositivo móvel que não só controlam o carro como também mostra a informação recolhida pelos vários sensores mostrando essa informação ao utilizador.

Irei dividir o trabalho em três partes:

- Descrição do material usado: irei descrever, ainda muito resumidamente, cada material utilizado
- Descrição das funcionalidades: depois de descrito cada componente, neste ponto irei descrever a sua utilidade para este projeto
- Física inerente ao projeto: Irei abordar a física que está aplicada a cada componente (uma vez que com este trabalho pretende-se aplicar os conhecimentos obtidos nas aulas (e alguma pesquisa fora das mesmas))

No final espero atingir os objetivos que me proponho, bem como colmatar as lacunas que apresento em algumas matérias da disciplina.

1. Descrição do material

Durante a realização do projeto utilizei vários materiais (sensores, módulos) afim de saber qual (ou quais) os mais indicados. Após alguma pesquisa e alguns testes encontrei os sensores que precisava e que passarei a descrever.

1.1. Plataforma de desenvolvimento

No mercado existe muitas ferramentas de prototipagem, microcontroladores, microcomputadores, uma panóplia de ferramentas que, para quem está a iniciar no mundo da eletrónica/robótica gera confusão e indecisão na hora de escolher a sua plataforma para trabalhar. Dos conhecimentos prévios que tinha (e depois de alguma pesquisa comparativa entre as plataformas) as opções para este projeto resumiam-se a duas plataformas: Arduino ou Raspberry Pi. Para perceber qual das duas a mais indicada fiz uma tabela de comparação:

Arduino UNO R3	**Raspberry Pi Model B**
• Microcontrolador • Memória: 32kb (0.5 usado no bootloader) • Processador: ATmega328P • Clock Speed: 16MHz • Pinos Digitais: 13 (6 com PWM) • Pinos Analógicos: 6 • Voltage: 5v • Input Voltage: 7 – 12v • Input Voltage (limit): 6 – 20v	• Microcomputador • Memória: 521MB • CPU: 700MHz • GPU: VideoCore IV a 250MHz • Saída vídeo/áudio • 26 pinos GPIO • 2 portas USB • Placa rede ethernet • Consumo: 700mA (3.5 W)

Após a analise comparativa das duas plataformas cheguei à conclusão que para a realização deste projeto necessito de um microcontrolador. Embora o Raspberry Pi tenha componentes mais sofisticados (ou mais evoluídos/atualizados digamos assim) que o Arduino, este último é o mais indicado para este tipo de trabalho.

1.2. Sensores

Para podermos determinar/calcular uma distância, medir qualquer coisa é necessário recorrer aos sensores. Tal como o corpo humano tem os seus sensores (audição, olfato, paladar, etc) na eletrónica/robótica é necessário recorrer a sensores externos para se obter informações que pretendemos. Neste projeto utilizei os seguintes sensores, sendo descritos com mais detalhes no capítulo 3 (Funcionalidades).

1.2.1. Sensor Distância HC-SR04

Este sensor permite calcular distâncias usando ultrassons. Na sua composição, este sensor tem 4 pinos de ligação:

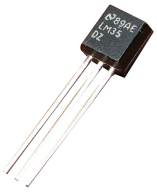
- Tensão (vcc – 5v)
- Terra (gnd)
- Emissor (trig)
- Recetor (echo)

De uma forma geral, este sensor emite um ultrassom (através do trig) que ao embater contra um obstáculo à sua frente é refletido e registado pelo recetor (echo). Pode medir até aos 400cm, tem um ângulo de precisão até aos 15º, sendo que tem uma imprecisão inicial de 2cm (a medição até aos 2cm iniciais é bastante imprecisa e apresenta valores irreais).

1.2.2. Sensor Temperatura LM35

Com este sensor é possível determinar a temperatura ambiente onde o carro se encontra. A sua ligação ao Arduino é feita através de três pinos:

- Tensão (vcc – 5v)
- Terra (gnd)
- Vout (ligação analógica)



Este pequeno sensor pode medir temperaturas entre os -55º até aos +150º, tendo uma precisão de 0,5º até aos 25º C.

1.2.3. Sensor de Luminosidade (LDR)

O sensor de luminosidade (LDR – **L**ight **D**ependent **R**esistor) é na verdade um tipo de resistência que varia consoante a luz existente. O seu funcionamento em termos gerais é se a luz for muito intensa, é diminuída a resistência, logo a corrente consegue fluir com mais facilidade, caso contrario, a resistência aumenta e a corrente não consegue fluir com tanta facilidade.

1.2.4. Buzzer

O buzzer é o responsável pela informação sonora do carro. O seu funcionamento é através de frequências (em Hertz) e do tempo (em milissegundos).

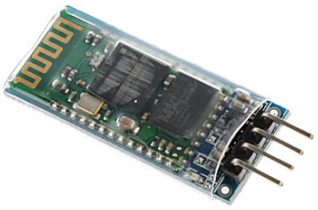
1.3. Módulos

Tal como os sensores, os módulos acrescentam mais funcionalidades ao nosso projeto. Abaixo irei descrever cada módulo, deixando a descrição física mais detalhada para o capítulo 4 (Física no Projeto).

1.3.1. Bluetooth HC-06

Havia várias formas de controlar o nosso carro, pelo que optei pela tecnologia bluetooth. Este módulo funciona apenas em modo escravo (slave), o seu alcance é de 10m e tem 4 pinos de ligação ao Arduino:

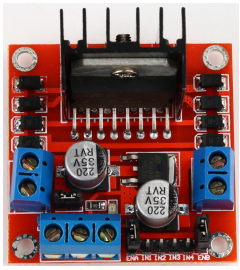
- Rx (comunicação recebida)
- Tx (comunicação emitida)
- Tensão (vcc – 5v)
- Terra (gnd)



1.3.2. Ponte H

A ponte H é um módulo que permite controlar até dois motores DC. Este módulo serve para controlar cargas indutivas (como por exemplo, relés, motores DC, etc), permite também o controlo da rotação do motor e velocidade, usando para isso os pinos com PWM[1]. São módulos bastante usados na robótica porque apresentam um circuito básico integrado e são de pequenas dimensões pelo que facilmente se coloca no projeto que se está a desenvolver. Os módulos utilizados neste projeto têm as seguintes características:

- Tensão para o motores: 5 – 35v
- Corrente máxima para os motores: 2A
- Potência máxima: 25W
- Tensão: 5v
- Corrente: 0 – 36mA



[1] Pulse Width Modulation

1.3.3. Motor DC

O Motor DC é o responsável pelo movimento das rodas, ou seja, o Motor DC faz girar as rodas. É um motor de corrente contínua de baixa tensão onde a rotação é baseado em campos magnéticos que passam nas suas bobines. Podemos controlar a velocidade através da tensão aplicada às mesmas. Para aplicar essa variação utilizei os pinos com PWM no Arduino.

1.4. Componentes

Para além dos sensores, módulos existem outros componentes que quando interligados entre si melhoram/recriam funcionalidades ao nosso projeto. Por exemplo, para ligar os vários sensores é necessário recorrer a uma placa designada de breadboard onde irão ser ligados todos os sensores, fios e demais materiais inerentes ao trabalho que estamos a realizar. Abaixo irei descrever os componentes que usei.

1.4.1. Breadboard

A breadboard (designada também por placa de ensaio ou por matriz de ensaio) é uma placa com furos e conexões condutoras que permitem a montagem de circuitos elétricos. São placas muito utilizadas em projetos de eletrónica por não ser necessário recorrer a soldaduras para se utilizar. Estas placas podem variar até aos 6000 furos e contêm, todas, conexões horizontais e verticais. Por norma, esta placa divide-se em três zonas: Zona de Superior, Zona Inferior e Zona Central. Nas duas extremidades (superior e inferior), existem duas linhas (com o sinal de + (positivo) e – (negativo)) onde as conexões são feitas na horizontal, sendo que a zona central as ligações são feitas na vertical não tendo restrições de conexões.

1.4.2. Fios / Cabos patch / Jumpers

Este é um componente que permite fazer a ligação (ponte) entre o componente e a breadboard e/ou Arduino, pois permite não só as ligações de energia como as ligações para a passagem dos dados/informação1 entre os mesmos.

1.4.3. Resistências

As resistências são um tipo de material que geram oposição à passagem da corrente elétrica, ou seja, a sua função serve para diminuir o fluxo elétrico num dado espaço. Cada resistência tem o seu valor, a sua “força”. Consoante o circuito que queremos fazer (ou temos quase pronto) assim é necessário determinar/calcular qual a melhor resistência a aplicar (irei descrever este cálculo no capítulo 4).

1.4.4. LED

O LED (Light Emitting Diode – Diodo Emissor de Luz) é um componente utilizado quando queremos emitir informação visual, ou seja, é um componente que emite uma luz. Os LEDs podem ser simples (mais comum de 5mm de diâmetro de várias cores) até aos mais complexos LED RGB (conseguem recriar várias cores utilizando a combinação de cores RGB (Red, Green, Blue – Vermelho, Verde e Azul)). Em termos de funcionamento, os LEDs operam a uma tensão que pode variar dos 1,6v até aos 3,3v.

1.5. Algoritmo

Embora o utilizador tenha que dar instruções ao carro para fazer qualquer tipo de atividade, o mesmo tem uma pequena inteligência artificial (doravante designada por IA). Esta IA está presente em dois momentos:

1. **Quando o veículo faz o movimento para a frente**: Neste caso, ao mover-se para frente, vai calculando a distância que o objeto à sua frente se encontra. Enquanto que o objeto for maior que 50cm de distância o movimento continua a mover-se, caso contrário o carro interrompe automaticamente o movimento e pára. Aqui o movimento só pode ser um dos três possíveis: para trás, esquerda ou direita.
2. **Acender LEDs**: O carro tem a capacidade de perceber se há (ou não) luz suficiente no local onde se encontra. Se no local onde se encontra houver muita luz é permitido ao utilizador, através da aplicação Android, acender/desligar os LEDs. Se a luz ambiente for fraca e/ou inexistente, o carro acende automaticamente os LEDs e não permite que o utilizador as desligue através da aplicação.

2. Funcionalidades

Depois de juntar os vários componentes surgiu a pergunta “o que fazer/implementar no carro?”. As funcionalidades permitem a interação com o utilizador, não só nas reações que o carro tem aos comandos como também, informa o mesmo com várias informações recolhidas dos vários sensores.

2.1. Medir distâncias

Esta funcionalidade permite ao carro evitar a colisão frontal, ou seja, através do sensor de ultrassons é calculada a distância do carro ao objeto que se encontra à sua frente, caso o objeto se encontrar a menos de 50cm o carro para e não permite mover-se para a frente (apenas para os outros três sentidos – para trás, esquerda ou direita).

Para determinar qual a distância recorremos à seguinte função:

Distância = (tempo ECHO no estado HIGH * Velocidade Som) / 2

Para se proceder à medição temos que colocar o emissor (pino trig) no estado HIGH por mais de 10µs (microsegundos). Depois, o emissor envia um ultrassom que ao embater no obstáculo é refletido e registado pelo recetor (pino echo). Durante este tempo o pino recetor ficará no estado HIGH. Com isto podemos determinar o primeiro parâmetro da função (**tempo ECHO no estado HIGH**), ou seja, o seu valor é registado de acordo com o tempo em que o pino recetor (echo) permaneceu no estado HIGH após o pino emissor ter sido colocado também no estado HIGH. Na programação em Arduino a função responsável por esta cálculo é a função *pulseIn*.

Para o segundo parâmetro (**Velocidade Som**) pretendemos guardar o valor numa escala métrica (em cm) daí ser necessário recorrer à velocidade do som. A velocidade podemos é considerada idealmente a 340m/s, logo o seu resultado será de metros por segundo (se considerarmos o tempo em segundos).

Para finalizar, a formula contém uma divisão por 2 uma vez que o ultrassom é enviado e refletido, logo percorre duas vezes o mesmo caminho**.**

2.2. Registar temperatura ambiente

Esta função permite registar a temperatura ambiente onde o carro se encontra. Este sensor apresenta uma saída de tensão linear proporcional à temperatura a que se encontra, sendo essa tensão varável de 0v até aos 5v. Existe outro detalhe, este sensor para poder ser “lido” pelo Arduino envia ao mesmo valores compreendidos entre o 0 e o 1023 (irei abordar com mais detalhe no próximo capítulo).

Para determinar a temperatura usei as seguinte expressão:

```
tempC = ( 5.0 * analogRead(analogSensorLM35) * 100.0 ) / 1023.0;
```

Para calcular a temperatura é lido o valor que está a ser lido pelo pino onde o sensor está ligado (neste caso, através da variável "analogSensorLM35"). Com este valor é adicionado a outro cálculo que determina o valor da temperatura. Como a porta analógica lê apenas de 0 a 1023 é necessário recorrer à divisão pelo valor máximo da porta para determinar a temperatura final.

2.3. Calcular luminosidade

Uma funcionalidade que achei interessante foi recriar aquilo que conhecemos dos candeeiros que se encontram na rua. Ou seja, queria com que o carro acendesse os LED's caso não houvesse luz suficiente no local onde se encontra. Pesquisei e encontrei um sensor (que na verdade é mais uma resistência) que permite fazer o que pretendia.

O resultado da pesquisa foi o LDR (Light Dependent Resistor – Resistor Dependente da Luz, ou também designado por foto resistor), sendo o responsável por tal tarefa. A sua função é aumentar/diminuir a sua resistência de acordo com a luz que incida sobre ele. Neste projeto coloquei o LDR um pouco afastado dos LED's para que estes não interfiram muito na recolha do valor da intensidade da luz no sensor.

Para o cálculo do valor da luz que incide sobre o sensor usei a seguinte função:

```
void lightSensor() {
  sensorLDRReading = analogRead(analogSensorLDR);

  if (sensorLDRReading < LDRMaxValueToTurnOnLED ) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Na função é proveniente lido o valor do pino analógico onde o sensor está ligado, valor esse que é comparado a uma constante previamente definida. Se o valor lido pelo sensor for menor que o valor da constante, os LED's acendem, caso contrário, não acendem (ou desliga – caso o estado anterior do LED).

2.4. Movimento

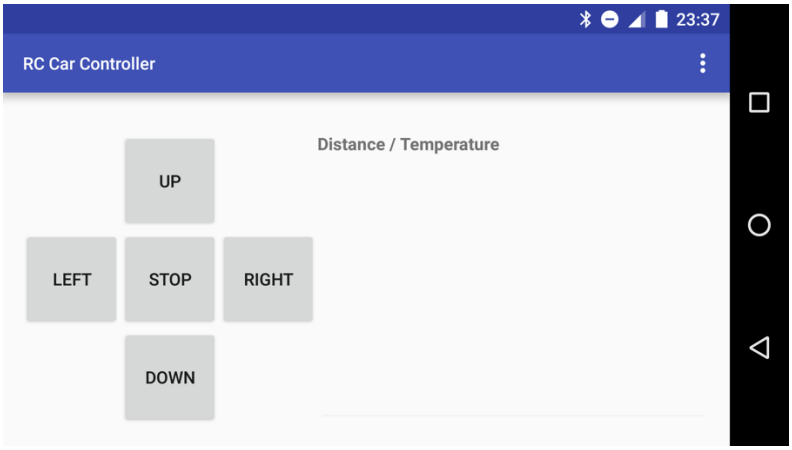
Como este projeto pretende recriar um carro de controlo remoto uma das funcionalidades básicas é dar movimento ao carro. O movimento é feito pela junção dos componentes Ponte H, Motor DC e Rodas, sendo gerado utilizando os pinos de entrada da ponte H (irei detalhar no próximo capítulo). No carro são permitidos os 4 movimentos padrão (frente, trás, esquerda, direita) e ainda adicionei o de "parar".

2.5. Buzinar

Tal como nos carros que conhecemos, uma outra funcionalidade que não podia deixar de lado era o poder buzinar. A buzina é feita através do sensor buzzer que permite a emissão de sinais sonoros usando uma frequência e a duração da mesma. Nesta funcionalidade o carro toca as primeiras notas do tema do jogo Super Mário.

2.6. Controlo Remoto

O controlo do carro faz-se através de um dispositivo Android (versão 5 ou superior) conectado usando a tecnologia Bluetooth. Segue uma imagem de como é a aplicação:



A aplicação poder-se-á dividir em três zonas:

- **Zona dos controlos do carro:** Local onde estão os botões para mover o carro
- **Zona das informações:** Local onde é mostrado as informações vindas do Arduino, tal como a distância do carro ao obstáculo que está à sua frente e a temperatura ambiente
- **Opções:** Onde se tem acesso às opções e/ou outras funcionalidades do carro que não são usadas tão frequentemente (tais como buzinar, acender/apagar os LED's, salvar os dados, etc)

2.7. Acender luzes

Como foi falado no ponto 3.3, o carro permite acender os LEDs de forma manual (menu opções, (Des)Ligar LED) ao invés do modo automático. No entanto só será possível caso a luz ambiente for superior a um certo valor, caso contrário o carro acende automaticamente os LED's e não permite desliga-los.

3. Física no Projeto

Com o desenvolver do projeto fui trabalhando com vários sensores, módulos e, ao mesmo tempo, adquirindo conhecimento não só das suas ligações mas como também como funcionam e como a física está presente nos mesmos. Este capítulo aborda essencialmente a física inerente a cada componente que utilizei na realização deste projeto.

3.1. Arduino: Pinos Analógicos e Pinos Digitais

O Arduino tem embutido dois tipos de pinos: Analógicos e Digitais. Cada pino tem o seu próprio comportamento.

Os pinos digitais têm apenas dois estados: Ligados (high - alto) ou Desligados (low - baixo), podendo ser representados em binário com 1 e 0 (respetivamente). Em termos de leitura de dados podemos considerar o pino no estado alto se a tensão variar entre os 3v e os 5v, caso varie entre os 0v e o 1v consideramos o pino no estado baixo. Existe ainda a variação entre os 1v e os 3v, para esta variação o pino continua no estado que estava antes de entrar nesta variação.

Para determinar com mais precisão os dados recebidos no Arduino utilizamos os pinos analógicos. Esta leitura permite ao Arduino traduzir a tensão recebida em valores que variam de 0 a 1023. Para esta tradução é necessário recorrer a um circuito interno chamado CAD (Conversor Analógico Digital) onde o resultado dessa conversão é um conjunto de 10bits (2^{10} = 1024 entradas).

No caso deste projeto, foi utilizado um Arduino UNO que possui 6 portas analógicas. Para utiliza-las, ligamos um sensor (no meu caso ou o LM35 (temperatura) ou o foto resistor (LDR)) a um dos pinos, definimos o pino na função Setup. No decorrer no programa (dentro da função loop (e/ou noutros métodos que irão ser chamados no loop) utilizamos a função “analogRead(pino)” para ler os valores.

3.1.1. PWM: Pulse Width Modulation

Uma das funcionalidades que os pinos digitais têm é a implementação do PWM (Pulse-Width Modulation ou Modulação por Largura de Pulso). O PWM é uma técnica de enviar mensagens/Informação através dos impulsos dos sinais, ou seja, através da largura de pulso de uma onda quadrada é possível controlar a potência e/ou velocidade.

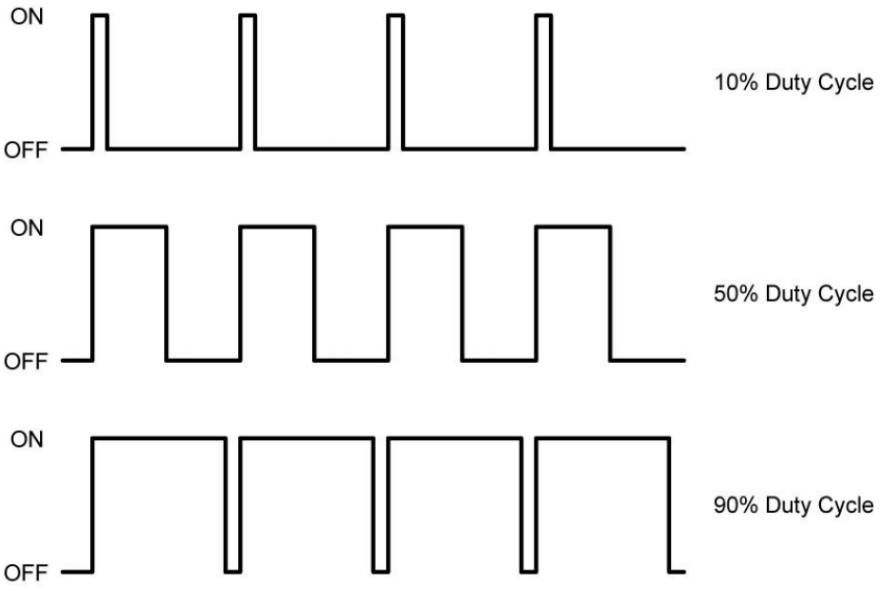
Como funciona?
Se considerarmos uma onda quadrada, para o bom funcionamento do PWM devemos variar a largura de pulso da onda, para efeito de cálculo é necessário recorrer a dois parâmetros: período (tempo) e a largura do pulso (chamado *duty cycle*). Observemos a sua expressão:

$$DutyCycle = 100X\frac{LarguraDoPulso}{Periodo}$$

Da sua expressão obtemos:

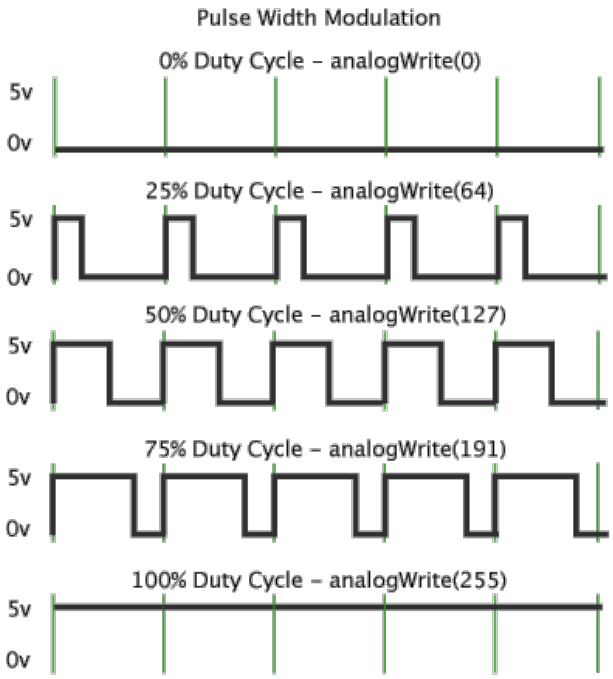
- **Duty Cycle:** Valor em percentagem
- **Largura de Pulso:** Tempo em que o sinal está ligado
- **Período:** Tempo do ciclo da onda

Abaixo estão alguns exemplos:



No caso do Arduino e neste projeto em concreto o uso do PWM permite controlar a velocidade e potência aplicada às rodas (e assim dar movimento ao carro).
Para escrever um sinal no pino com PWM é necessário recorrer à função "analogWrite" do Arduino que recebe como parâmetros o pino com PWM e o valor do Duty Cycle, sendo este último guardado em 8bits de forma a que o valor esteja compreendido entre 0 (0% de Duty Cycle) e os 255 (100% de Duty Cycle). Um exemplo seria:

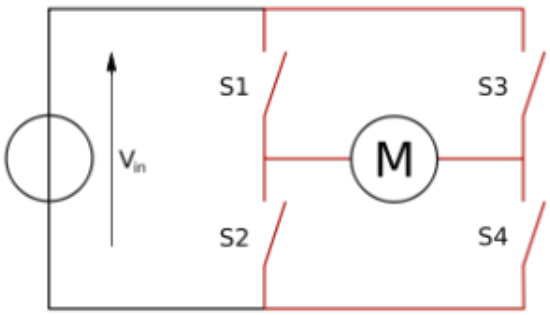
```
analogWrite(motorAAtrasIN1, 64);
analogWrite(motorAAtrasIN2, 127);
analogWrite(motorAAtrasIN2, 191);
analogWrite(motorAAtrasIN2, 255);
```



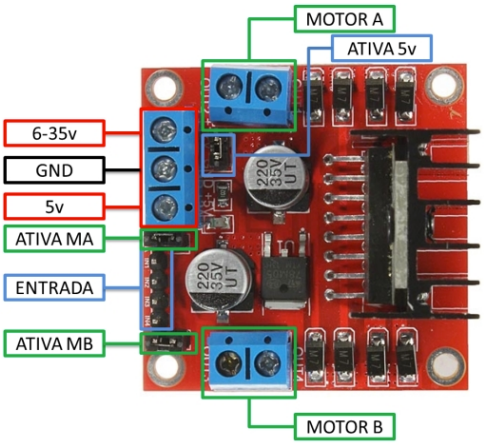
Quando se altera o Duty Cycle altera-se também o valor médio da onda pelo que, se utilizarmos uma ferramenta para observação de ondas (um osciloscópio), a onda obtida pelo PWM é um sinal com amplitude constante e de valor igual ao valor médio da onda.

3.2. Ponte H

O uso da Ponte H permite não só fornecer aos motores DC a corrente necessária como também inverter o seu sentido giratório através das operações de um microcontrolador. É constituída por quatro "chaves" (S1 – S4) que são acionadas de forma alternada (S1 e S4 ou S2 e S3). Segue um exemplo ilustrativo:



Para este projeto utilizei as Ponte H L298N porque apresentam um menor espaço ocupado, baixa complexidade do circuito e possui alguns elementos importantes integrados, como por exemplo um regulador de intensidade. A ponte é dividida nas seguintes partes:



- **Motor A/B:** Onde são conectados os motores DC
- **Ativa MA/MB:** Estes pinos são os responsáveis pelo controlo PWM dos motores A/B. Se os pinos estiverem com jumper não haverá controlo de velocidade, pois estarão ligados aos 5v. Estes pinos podem ser utilizados em conjunto com os pinos PWM do Arduino
- **5v:** Quando a ponte estiver a funcionar entre os 6 – 35v, o regulador de intensidade disponibiliza uma saída regulada de +5v no pino (5v) para um uso externo (com jumper), podendo alimentar, por exemplo, outro componente. No entanto não podemos alimentar este pino com +5v do Arduino se estivermos a controlar um motor de 6-35v com o jumper conectado, isto irá danificar a ponte. Este pino somente se torna numa entrada caso se esteja a controlar um motor de 4 – 5,5v (sem jumper), assim podemos utilizar a saída +5v do Arduino.
- **6 – 35v e GND:** A estes pinos conectados uma fonte de alimentação externa (no caso de se utilizar duas pontes a segunda ponte tem como fonte externa a primeira) quando temos que controlar um motor que opere entre 6-35v. Se o nosso motor DC utilizar 12v, é aqui que é conectada a fonte externa de 12v.
- **Entrada:** Aqui estão quatro pinos (IN1-4) que são responsáveis pela rotação do motor, ou seja, para o motor A temos os pinos IN1 e IN2, para o motor B os pinos IN3 e IN4)

Na tabela abaixo mostra a ordem de como os motores são ativados, neste caso para o motor A (para o motor B o esquema é o mesmo mas para os pinos IN3 e IN4):

MOTOR	IN1	IN2
Sentido Horário	5v	GND
Sentido Anti-Horário	GND	5v
Parado	5v	5v

Falei no PWM no ponto anterior e voltei a referir que as pontes H utilizavam também o PWM para controlar os motores, mas “Como se faz?” é a pergunta que fica no ar e que irei responder.

Para utilizar o PWM nas pontes temos que retirar o jumper do Ativa MA/MB para que a tensão seja modulada e enviada para o motor no mesmo formato. Isto ocorre porque a ponte H só irá "funcionar" enquanto o sinal de enable estiver com 5v.

Assim sendo, para um motor DC será um sinal PWM com um duty Cycle igual ao do enable e será calculada a tensão média da seguinte forma:

$$V_{med} = V_{max}(\text{Tensao Ponte H}) * \text{Duty Cycle (\%)}$$

3.3. Sensor Temperatura LM 35

Como referi no ponto 3.2, este sensor apresenta uma tensão linear relativa à temperatura a que ele se encontra, apresentado na sua saída um sinal de 10mV para cada grau Celcius, não sendo necessário recorrer a nenhuma calibração.

Apresenta três conexões, uma para o GND (groud – ou terra), outra para a alimentação (vcc – 5v) e outra para a transmissão dos dados.
O seu funcionamento é bastante simples, para cada 10mV de saída representa um grau Celcius. Quando se usa os 5v, o resultado é de 5000mv/1023 = 4.8mV.

3.4. Motor DC

O motor DC é um motor de corrente continua que converte a corrente elétrica em energia mecânica, ou seja, é um mecanismo que ao receber a corrente elétrica faz girar os mecanismos existentes no interior do motor. É bastante utilizado porque reúne características que o torna vantajoso o seu uso, tal como construção simples, custo reduzido, facilidade de transporte e simplicidade de comando/operação.

O motor é constituído por duas bobines que formam uma força magnética que permite efetuar a rotação do motor com velocidade angular constante.

Neste trabalho os motores apresentam as seguintes características:

Tensão	3v	5v	6v
Corrente	100mA	100mA	120mA
RPM (com roda+pneu)	100	190	240
Velocidade (m/s)	20	39	48
Tamanho	48:1		

3.5. LDR

Como referi anteriormente o LDR é um tipo de resistência que variam de acordo com a luminosidade do ambiente em que estão. Através do circuito que está presente no

projeto (vide anexo III) podemos calcular o valor da tensão à saída através da seguinte fórmula:

$$Vsaída = \frac{10000}{10000 + RLDR} * 5$$

Pegando em exemplos, se o ambiente onde o carro está apresentar um ambiente quase escuro, o LDR apresenta cerca de 500kΩ e o valor da tensão será:

$$Vsaída = \frac{10000}{10000 + 500000} * 5 = 0{,}098V$$

No lado oposto, se o ambiente apresentar um ambiente claro, com bastante luminosidade, o LDR apresenta cerca de 300Ω, sendo a sua tensão calculada da seguinte forma:

$$Vsaída = \frac{10000}{10000 + 300} * 5 = 4{,}85V$$

3.6. Resistências

A resistências têm um papel fundamental para o equilíbrio de um sistema elétrico, pois elas conseguem fazer um balanço da corrente para que os componentes recebam a corrente adequada ao seu bom funcionamento.

Como referi anteriormente, a função da resistência é criar uma "barreira" para que a corrente elétrica flua do modo esperado. O seu cálculo é efetuado através da Lei de Ohm (irei explicar no ponto abaixo) sendo o seu resultado em Ohm (Ω).

3.6.1. Lei de Ohm

A lei de Ohm foi criada pelo alemão George Ohm e refere que a razão entre a tensão (entre dois pontos) e a corrente elétrica é constante, sendo aplicada pela seguinte fórmula:

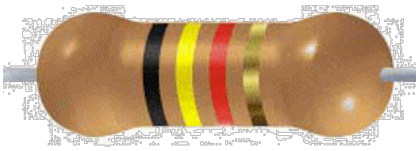
$$R = \frac{V}{I} \qquad \text{ou} \qquad V = R * I$$

onde:

- **R:** é a resistência medida em Ohm (Ω)
- **V:** é a diferença de potencial elétrico (ou tensão) medida em Volt (v)
- **I:** é a intensidade da corrente elétrica medida em Ampere (A)

3.6.2. Como calcular a resistência certa

O cálculo das resistências necessárias pode ser feito através de uma tabela de cores, cores essas que já vêm impressas na própria resistência. Uma resistência tem a seguinte forma:



Podemos observar que a resistência contem quatro bandas, de várias cores impressas. Cada banda equivale a um termo para determinar o seu valor. Para determinar o seu valor recorremos a uma fórmula, formula essa que são necessários quatro parâmetros:

- 1º Parâmetro: 1ª banda
- 2º Parâmetro: 2ª banda
- 3º Parâmetro: 3ª banda (multiplicador)
- 4º Parâmetro: 4ª banda (tolerância)

Para nos ajudar com as cores, guiamo-nos pela seguinte tabela de cores:

Cor	1ª Banda	2ª Banda	Multiplicador	Tolerância
Preto	0	0	X 10^0	
Castanho	1	1	X 10^1	+/- 1%
Vermelho	2	2	X 10^2	+/- 2%
Laranja	3	3	X 10^3	+/- 3%
Amarelo	4	4	X 10^4	+/- 4%
Verde	5	5	X 10^5	+/- 0,5%
Azul	6	6	X 10^6	+/- 0,25%
Roxo	7	7	X 10^7	+/- 0,1%
Cinzento	8	8	X 10^8	+/- 0,05%
Branco	9	9	X 10^9	
Dourado			X 0,1	+/- 5%
Prateado			X 0,01	+/- 10%

Nota: Existem resistências com mais bandas, no entanto as mais comuns para pequenos projetos de eletrónica e neste projeto em concreto é mais comum encontrar resistências de 4 bandas.

Para determinar qual o valor da resistência pegamos na formula de Lei de Ohm e calculamos da seguinte forma (utilizemos uma resistência presente neste projeto):

- 1ª Banda: Vermelho - 2 (1º algarismo)
- 2ª banda: Vermelho - 2 (2º algarismo)
- Multiplicador: Castanho - x10^1
- Tolerância: Dourado - $\pm$ 5%

Com as duas primeiras bandas, multiplicamo-las pelo valor do multiplicador correspondente. Para tal utilizemos a fórmula vista anteriormente pela Lei de Ohm:

$$R = \frac{V}{I} \quad \Leftrightarrow \quad R = V * I \quad \Leftrightarrow \quad R = 22 \times 10^{1} = 220\Omega$$

Há outra forma de determinar o valor da resistência, usando uma ferramenta, o voltímetro.

Conclusão

Findo a realização deste projeto os meus objetivos foram amplamente atingidos. No inicio referi que tinha lacunas nesta matéria de Física e que o “problema” que tinha muitas vezes era em não obter o material, mas sim “o que fazer” com o material, quando o terminei o conhecimento sobre este tema não só aumentou como não me sinto “preso” ao facto de não saber o que fazer com o material em mão.

A realização deste trabalho deu para perceber as potencialidades do Arduino, como esquematizar circuitos elétricos e posterior montagem, certos cuidados que devemos ter ao manusear componentes eletrónicos sensíveis.

Pós o término do projeto, vejo-o como uma boa base para futuros upgrades e/ou novas funcionalidades a juntar àquelas que já tem. Não descarto a hipótese de continuar este projeto num caminho de autodidata (aprender um pouco mais), como também no caminho de projeto final de curso.

Tentei aplicar ao máximo os conhecimentos obtidos nas aulas, bem como alargar conhecimentos nas diversas pesquisas que fiz ao longo da montagem do projeto. Sei que as lacunas não foram completamente suprimidas, mas reconheço que aprendi bastante com a realização deste trabalho. Penso que atingi os objetivos a que me propus inicialmente.

Webgrafia

http://www.gta.ufrj.br/grad/01_1/foto/fotoresistores.htm
http://www.gta.ufrj.br/grad/01_1/foto/fotoresistores.htm
https://pt.wikipedia.org/wiki/LDR
http://www.gta.ufrj.br/grad/01_1/contador555/ldr.htm
https://pt.wikipedia.org/wiki/Arduino
http://loja.vidadesilicio.com.br/pd-b2945-driver-motor-ponte-h-l298n.html?ct=9df63&p=1&s=1
http://domotica12d.blogspot.pt/2010/01/resistencia.html
http://forum.clubedohardware.com.br/topic/908549-resolvido-como-calcular-resistência-me-digam-cada-detalhezinho/
https://blog.udemy.com/arduino-ldr/
http://www.circuitstoday.com/pwm-generation-and-control-using-arduino
https://cdn.sparkfun.com/assets/resources/4/4/PWM_output_Arduino.pdf
http://nilsonmori.blogspot.pt/2011/05/principio-basico-de-funcionamento-de-um.html
http://www.mecaweb.com.br/eletronica/content/e_pwm
https://opensource.com/life/15/5/should-i-get-arduino-or-raspberry-pi
http://readwrite.com/2014/05/07/arduino-vs-raspberry-pi-projects-diy-platform
http://blog.vidadesilicio.com.br/arduino/basico/entradas-e-saidas-analogicas/
http://blog.vidadesilicio.com.br/arduino/ponte-h-l298n-controle-velocidade-motor/
http://blog.vidadesilicio.com.br/arduino/modulo-ponte-h-l298n-arduino/
https://www.youtube.com/watch?v=1G1Amb7kvZU
http://digitalhacksblog.blogspot.de/2012/05/arduino-to-android-turning-led-on-and.html
http://loja.vidadesilicio.com.br/pd-e0b8c-modulo-hc-06-bluetooth.html
http://solderer.tv/cxemcar/
https://github.com/cxemnet/CxemCar1
http://www.embarcados.com.br/arduino-saidas-pwm/

ANEXOS

ANEXO I – Código Arduino

Uma vez que o código é bastante extenso, coloquei-o num repositório online. Esse repositório pode ser acedido através do seguinte endereço:

https://github.com/MacgyverPT/Arduino-RC_Car/blob/master/Data/ArduinoSketch/ArduinoSketch.ino

Notas: O código encontra-se todo comentado.

ANEXO II – Código Android

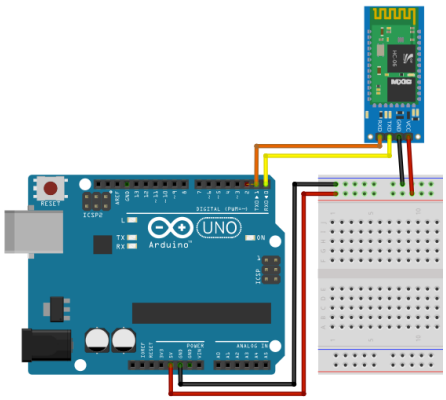
Uma vez que o código é bastante extenso, coloquei-o num repositório online. Esse repositório pode ser acedido através do seguinte endereço:

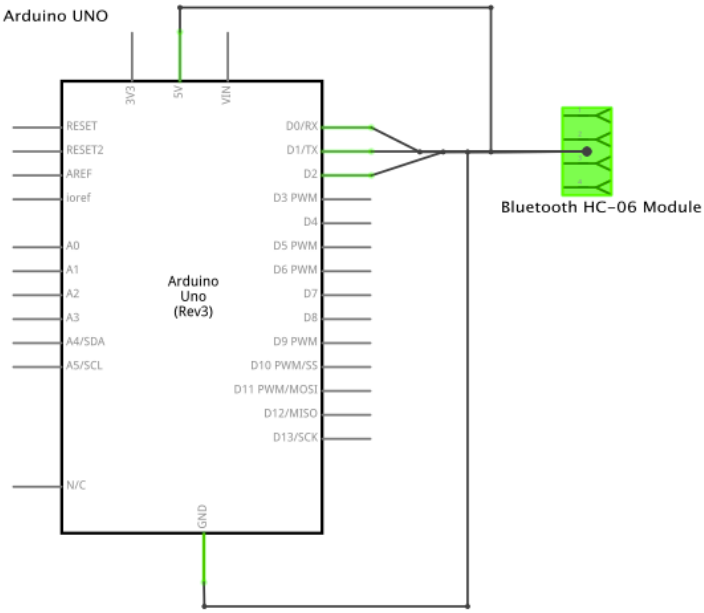
https://github.com/MacgyverPT/Arduino-RC_Car/tree/master/Applications/Android/app/src/main/java/rccar/android

ANEXO III – Esquemas circuitos

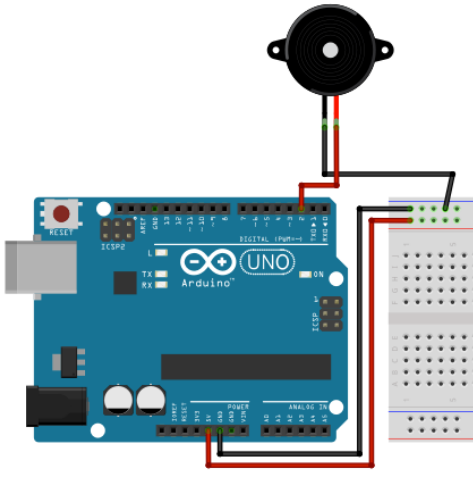
Tal como o código, está disponível no repositório as imagens com mais detalhe e os ficheiros das ligações feitos usando o software open source, Fritzing. O endereço é o seguinte: github.com/MacgyverPT/Arduino-RC_Car/blob/master/Data/Diagrams

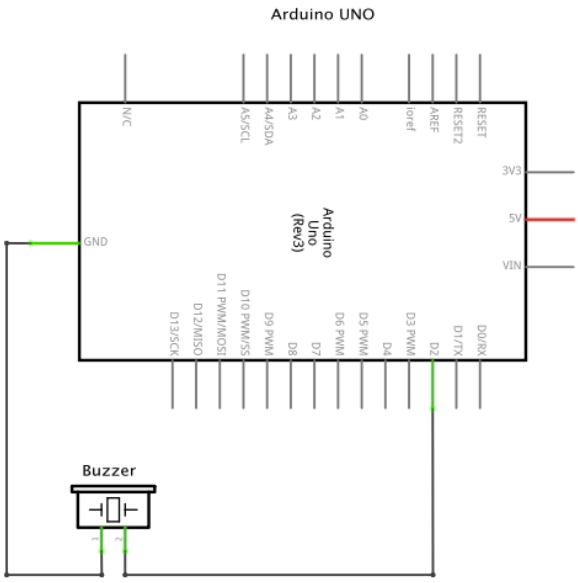
- **Bluetooth**



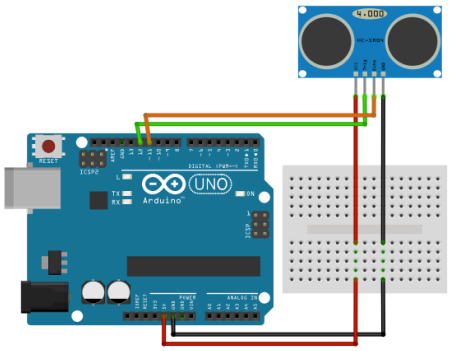


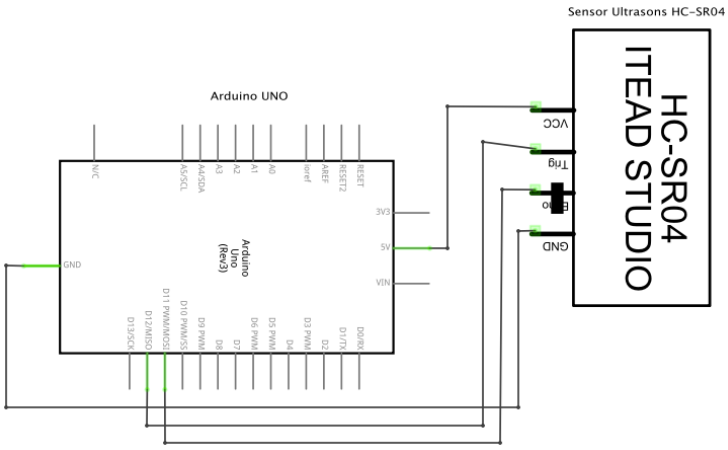
- **Buzzer**



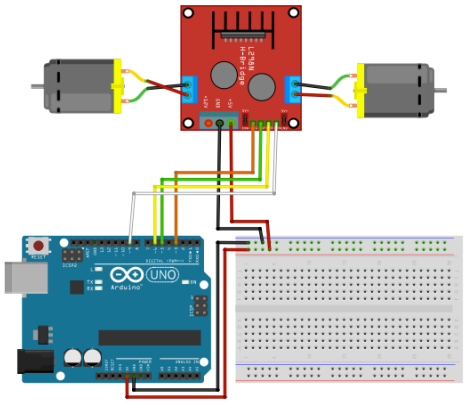


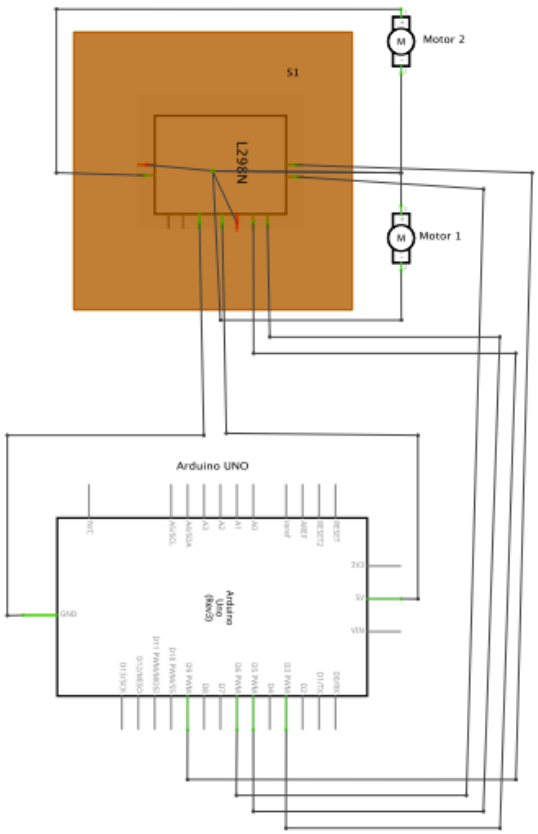
- **Sensor distâncias HC-SR04**

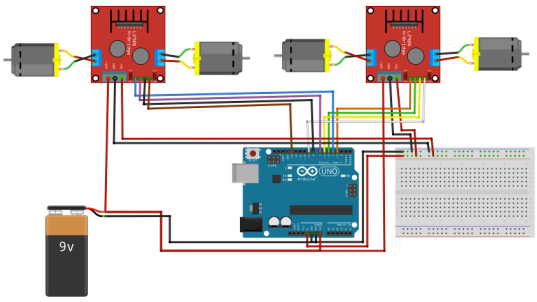


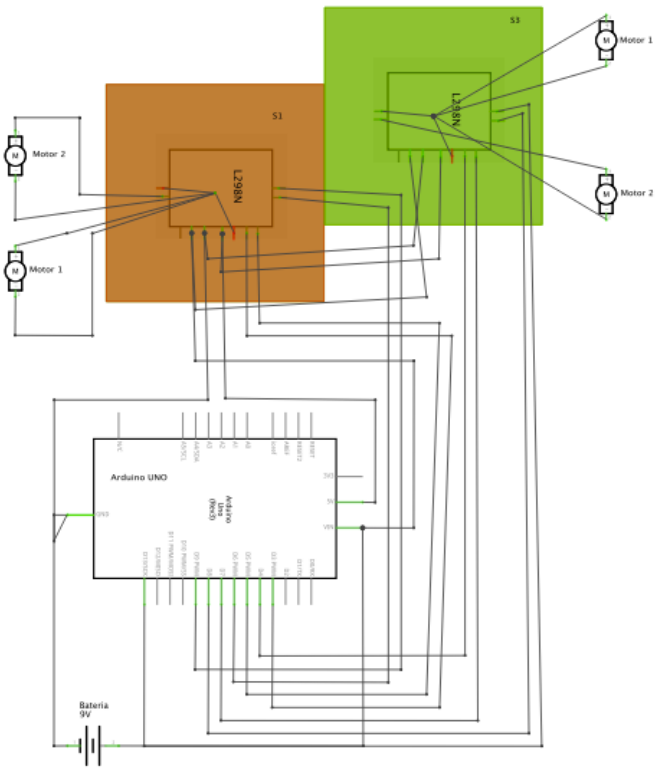


- **Ponte H**

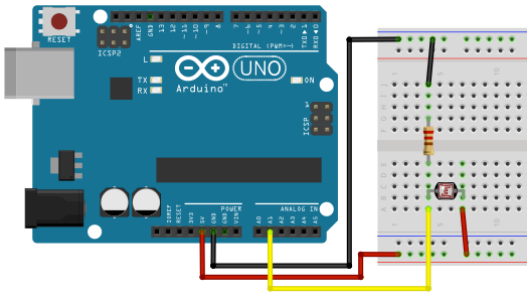


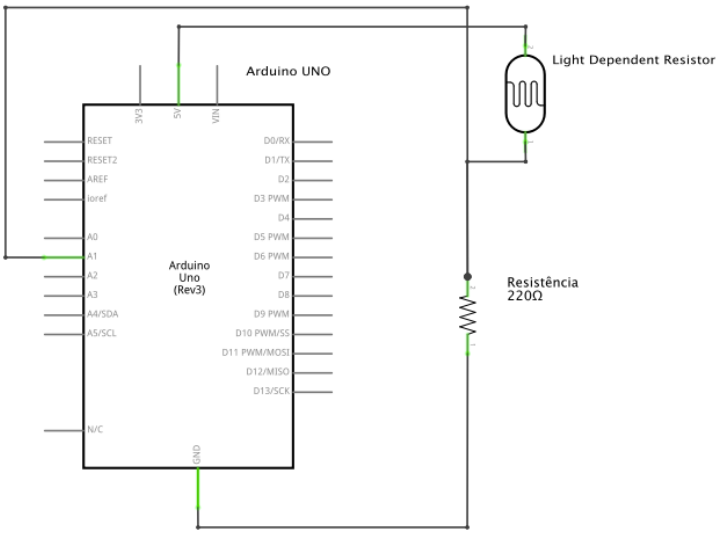



9v

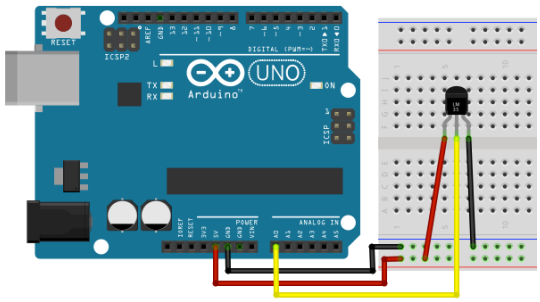

S1
L298N
Motor 1
Motor 2
Arduino UNO
Bateria
9V

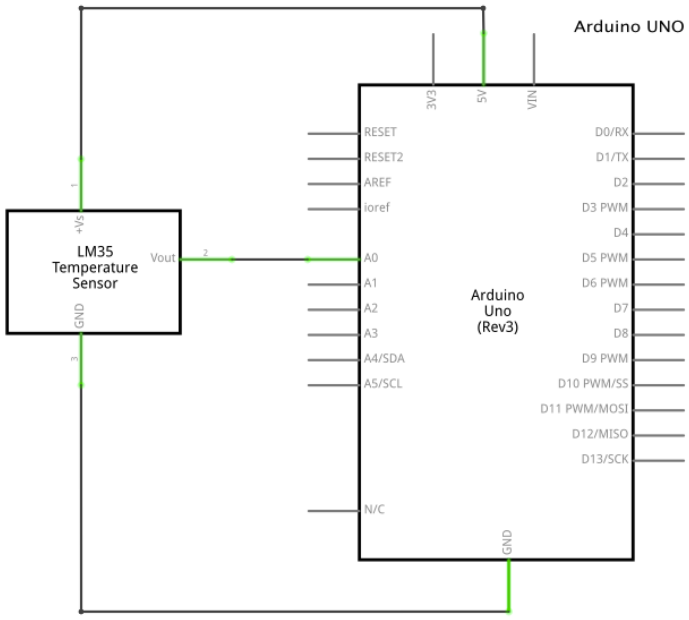
- **LDR**



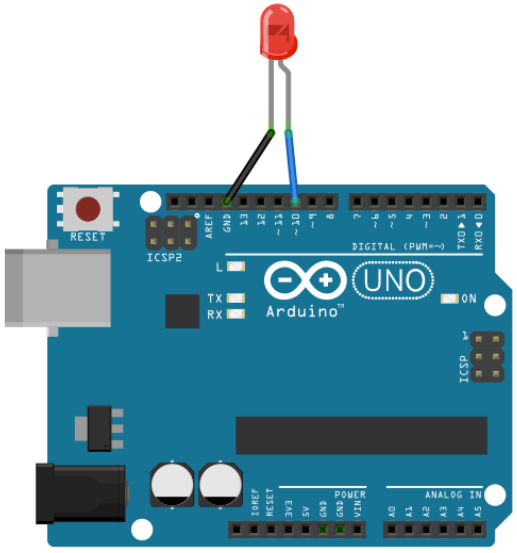

Arduino UNO
Light Dependent Resistor
Arduino
Uno
(Rev3)
Resistência
220Ω

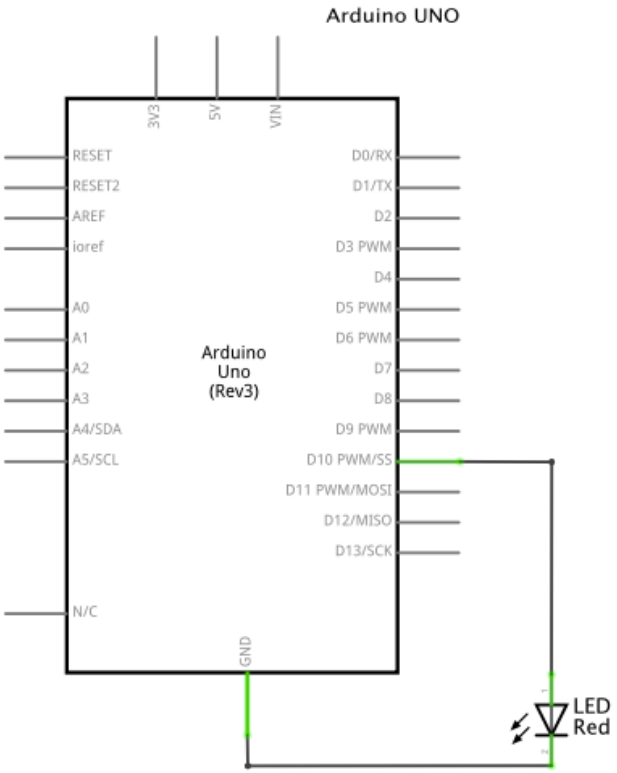
- **LM35**




Arduino UNO
3V3
5V
VIN
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
GND
LM35
Temperature
Sensor
Vout
+Vs
GND
Arduino
Uno
(Rev3)

- **LED**




Arduino UNO
3V3
5V
VIN
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
GND
Arduino
Uno
(Rev3)
LED
Red

ANEXO IV – Fotos da montagem

As fotos do processo de montagem encontram-se disponíveis a partir do seguinte endereço: http://tinyurl.com/Fotos-Montagem

ANEXO V – Vídeos de teste

Os vídeos de realização de testes deste projeto encontram-se no meu canal do youtube através do seguinte endereço:
https://www.youtube.com/playlist?list=PLzY7ljRsFjstPsBfRJprn8UKAdcKGBlr0