APÊNDICE 06: PRODUTO EDUCACIONAL

SERVIÇO PÚBLICO FEDERAL

UNIVERSIDADE FEDERAL DO SUL E SUDESTE DO PARÁ

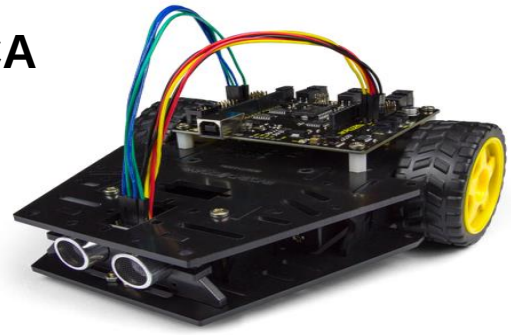
INSTITUTO DE CIÊNCIAS EXATAS – ICE

PROGRAMA DE PÓS´GRADUAÇÃO EM FÍSICA

MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA

ROTEIRO DE APLICAÇÃO PARA:

ENSINO DE ENERGIA MECÂNICA ATRAVÉS DA ROBÓTICA



ROGÉRIO RUIZ DO AMARAL

Marabá-Pa

2020

ESTRUTURA DO PRODUTO

1. INTRODUÇÃO

O ensino de física tem evoluídos no sentido das implementações de práticas pedagógica inovadoras e contextualizada. A tecnologia está cada vez mais acessível aos estudantes e professores, funcionando assim como um aliado em muitas frentes de ensino (Pereira, et.al., 2020, p.29).

Utilizando métodos e roteiros para aplicação dos conceitos de robótica, se percebe uma resposta significativa dos alunos, no que diz respeito à relação da prática e a teoria em um processo de construção de uma linguagem mais complexa, porém com aplicabilidade na Física e em outras áreas do conhecimento. Conteúdos como cinemática e dinâmica, podem ser trabalhados em um ambiente convidativo e interativo, fazendo com que o processo de ensino e aprendizagem tenha um desempenho e uma participação dos discentes com motivação diferenciada.

Dois kits serão utilizados nessa proposta afim de disponibilizar uma interface de possíveis comandos para auxiliar na construção do conhecimento teórico através de seu Robô falcon, também conhecido na grande rede como seguidor de linha, e também o Braço robótico, sendo possível com isso, o uso da prática para facilitar o processo de ensino e aprendizagem do aluno.

2. CONHECENDO O ROBÔ FALCON E A PLACA JULIETA

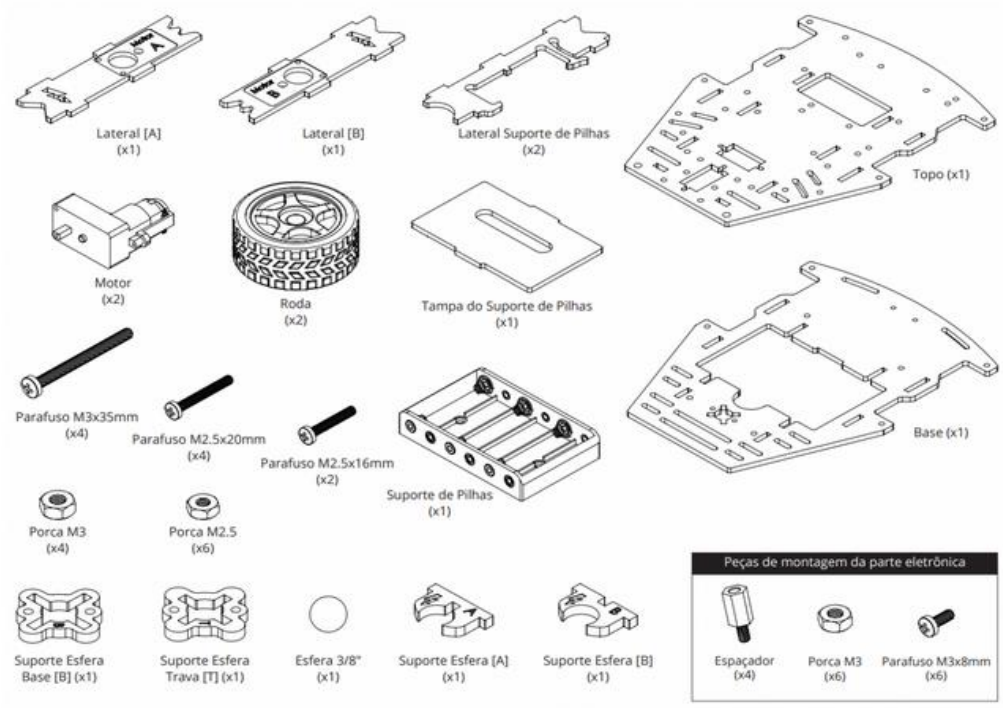
O Robô Falcon pode ser construído com matérias de baixo custo, porém é inevitável a compra de alguns itens para incluir a programação responsável pelas ações do robô. Esse robô pode ser considerado um dos mais interessantes para ser montado com alunos do ensino médio, pois apresenta muitas aplicações, dentre elas: elétrica, mecânica, sistema de medidas entre outras, e uma programação bem acessível, facilitando o acesso a esse universo de programação C++.

A estrutura mecânica deste Robô necessita de alguns itens, são eles:

- ✓ 01 x Plataforma Robótica Falcon
- ✓ 01 x Placa Julieta V1.0
- ✓ 01 x Sensor Ultrassônico HC-SR04
- ✓ 01 x Suporte para Sensor Ultrassônico
- ✓ 02 x Sensor de Refletância QRE - Analógico

- ✓ 06 x Pilhas AA
- ✓ 10 x Jumper Premium 20 cm F/F
- ✓ 01 x Rolo de Fita PVC para traçado
- ✓ 01 x Cabo USB AB

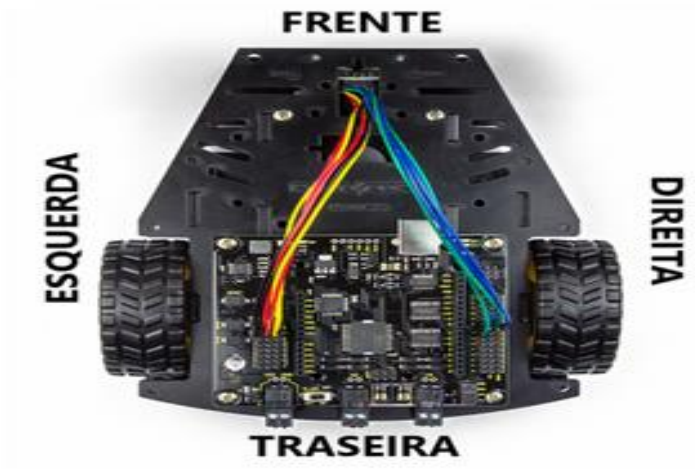
Figura 01: Itens da estrutura mecânica



Fonte: https://www.robocore.net/tutoriais/kit-iniciante-robotica-mecanica

Após a montagem da estrutura mecânica, que pode ser feita com ajuda de diversos tutoriais na internet ou até mesmo com a criatividade do próprio aluno, o Robô Falcon pode assumir a seguinte estrutura física:

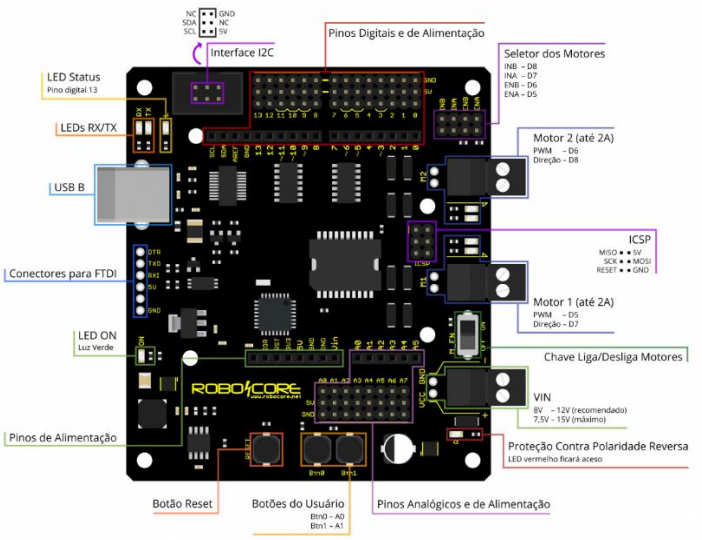
Figura 02: Estrutura mecânica



Fonte: https://www.robocore.net/tutoriais/kit-iniciante-robotica-mecanica

A movimentação e interação do Robô com o ambiente virtual necessita da parte elétrica e essa estrutura é montada com uma placa de arduino ou similar, no projeto em questão foi utilizada a placa Julieta. Essa opção está relacionada a alguns fatores, como o custo benefício e a versatilidade para aplicações na Física. A placa em questão apresenta um barramento de entrada e saída no padrão de 3 pinos, que permitem conectar sensores, servo motor e entre outros. Existe também um chip de Ponte-H integrado, responsável por controlar motores de forma livre com até 2ª de corrente elétrica em cada servo.

Figura 03: Placa Julieta



Fonte: https://www.robocore.net/tutoriais/kit-iniciante-robotica-mecanica

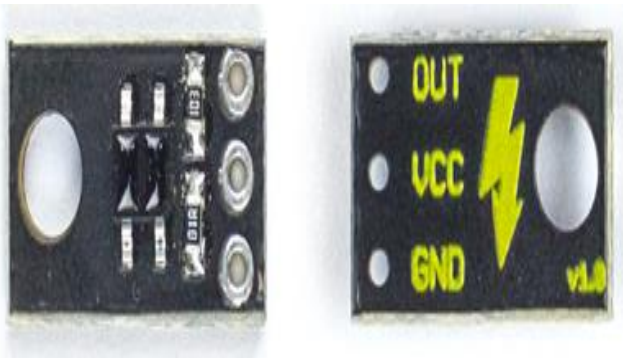
Afim de melhorar as informações sobre a Julieta, vejamos alguns itens importantes da placa:

- ✓ **USB B:** conector USB por onde a placa é programada e se comunica com o computador.
- ✓ **LEDs RX/TX:** indicam comunicação na porta USB. Normalmente piscam rapidamente quando a placa é reprogramada ou quando existe uma comunicação entre a placa e um computador.
- ✓ **LED Status:** conectado ao pino 13 da placa, esse LED pode indicar se o código foi carregado e se está rodando. Também é bastante utilizado para fins gerais.
- ✓ **Interface I2C:** conector no formato IDC, disponibiliza os pinos A4 (SDA) e A5 (SCL) permitindo a fáceis conexões com dispositivos I2C.
- ✓ **Pinos Digitais:** barramento que contém os pinos digitais da placa. Cada pino é disponibilizado em um conector de 3 pinos que contém também um pino de GND e 5V.
- ✓ **Seletor dos Motores:** jumpers responsáveis por conectar os pinos do micro controlador da Julieta aos pinos do chip de Ponte-H integrado (ENA - D5, ENB - D6, INA - D7 e INB - D8).
- ✓ **Motor 2:** saída para o motor 2. Pode acionar uma carga total de até 2A. Controlado pelos pinos D6 e D8 (PWM e Direção, respectivamente).
- ✓ **ICSP:** conector para gravação utilizando programador externo. Também disponibiliza os pinos do barramento SPI da placa.
- ✓ **Motor 1:** saída para o motor 1. Pode acionar uma carga total de até 2A. Controlado pelos pinos D5 e D7 (PWM e Direção, respectivamente).
- ✓ **Chave Liga/Desliga Motores:** chave de habilitação e desabilitação dos motores. **VIN:** entrada de tensão da bateria ou fonte de alimentação dos motores.
- ✓ **Proteção Contra Polaridade Reversa:** um led vermelho é aceso quando a fonte de alimentação é conectada com a polaridade invertida.
- ✓ **Pinos Analógicos:** barramento que contém os pinos de entrada analógica da placa. Cada pino é disponibilizado em um conector de 3 pinos que contém também um pino de GND e 5V.
- ✓ **Botões de Usuário:** dois botões para uso geral nos pinos A0 e A1. Deve-se habilitar os resistores de pull-up internos da placa para sua utilização.

- ✓ **Botão Reset:** retorna o fluxo do código na placa para o início quando pressionado. Este botão não apaga o código da placa.
- ✓ **Pinos de Alimentação:** disponibilizam pinos com GND, 3.3V, 5V e VIN para uso geral.
- ✓ **LED ON:** indica que a placa está energizada. Quando aceso, indica que a bateria/fonte ou o cabo USB está conectado.
- ✓ **Conector para FTDI:** permite utilização de um conversor USB-Serial externo caso o FTDI da placa seja danificado.

Outro item importante no Robô Falcon é o sensor de refletância QRE, que é responsável por detectar linhas. Esse sensor, figura 04, trabalha detectando a luz refletida oriunda do próprio LED infravermelho.

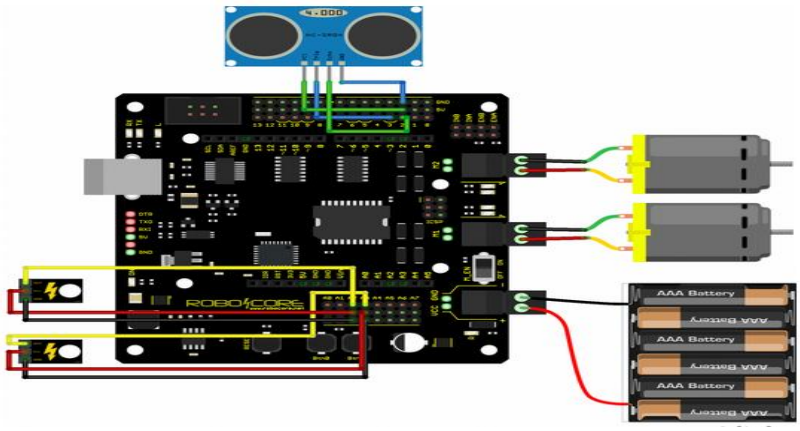
Figura 04: Sensor de Refletância QRE



Fonte: https://www.robocore.net/tutoriais/kit-iniciante-robotica-mecanica

Após ter conhecido os itens principais do Robô, é necessário realizar as ligações nas portas correspondentes. Para isso, basta conferir a imagem abaixo e ligar corretamente cada sensor e motores na placa Julieta.

Figura 05: Montagem da estrutura elétrica do Robô Falcon



Fonte: https://www.robocore.net/tutoriais/kit-iniciante-robotica-eletronica

Finalizado esse momento, o Robô pode começar a receber informações via USB, do ambiente virtual responsável por carregar as ações da placa. Esse software pode ser encontrado nos sites de busca pelo endereço https://www.arduino.cc/en/Main/Software, sendo gratuito e com atualizações leves que não comprometem o uso deste programa no computador ou notebook.

3. CONHECENDO O BRAÇO ROBÓTICO E A PLACA BLACK BOARD UNO R3

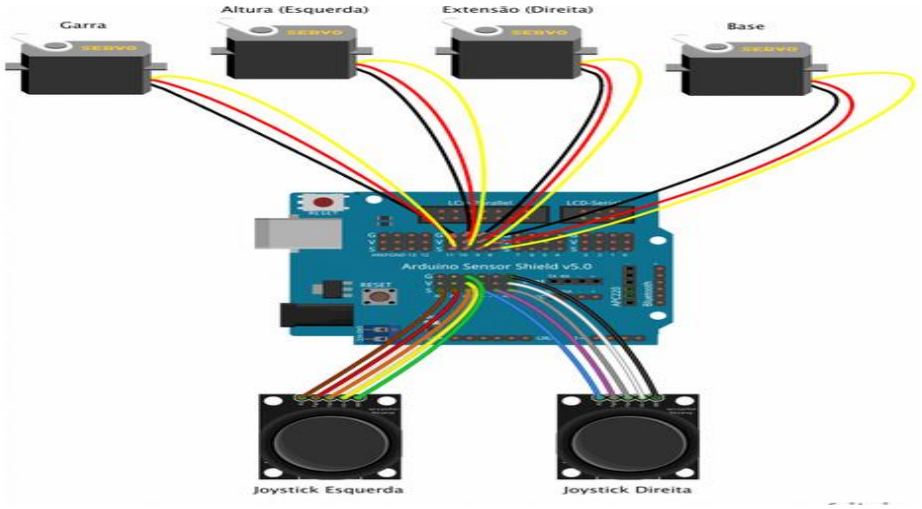
Com a intenção de introduzir outros conceitos de robótica associado a programação e mecânica, o Braço Robótico é um excelente artifício didático para aprimorar o aprendizado de Física e ao mesmo tempo estimular o aluno a interagir com a tecnologia.

As estruturas completas dos braços robóticos disponíveis no mercado necessitam, invariavelmente, de alguns itens, são eles:

- ✓ 01 Braço Robótico RoboARM
- ✓ 01 Placa Uno R3 com cabo USB
- ✓ 01 Sensor Shield V5
- ✓ 01 Controle BatPad
- ✓ 01 Fonte 5V 5A

Os Controles BadPad com seus respectivos jumpers são utilizados para fornecer ao aluno uma maior liberdade na condução do experimento. A imagem seguinte ilustra como deve ser ligado os servos e os controles na placa:

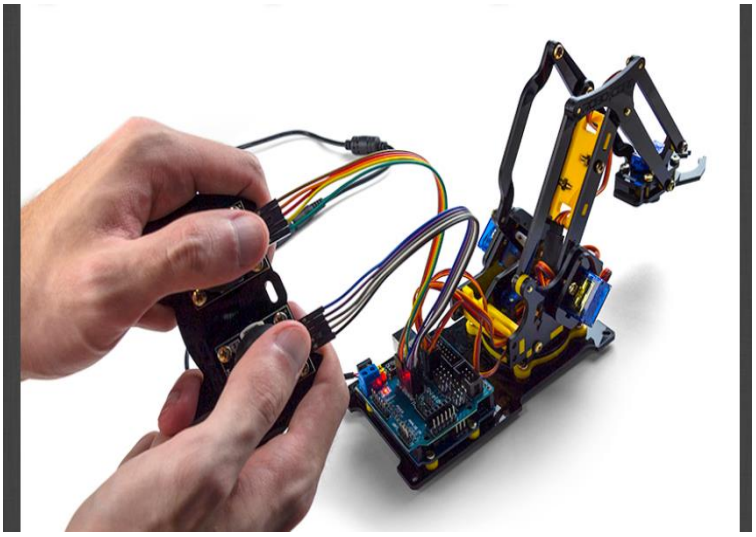
Figura 06: Arduino recebendo os servos e os controles joystick



Fonte: Retirada do site https://www.robocore.net/tutoriais/kit-iniciante-robotica-eletronica

A Figura 15 ilustra o primeiro item do projeto em questão, o RoboARM, ou seja, o braço, onde será conectado as placas, os sensores e o controle BatPad. Quando todos os itens estiverem conectados, a estrutura ficará da seguinte forma:

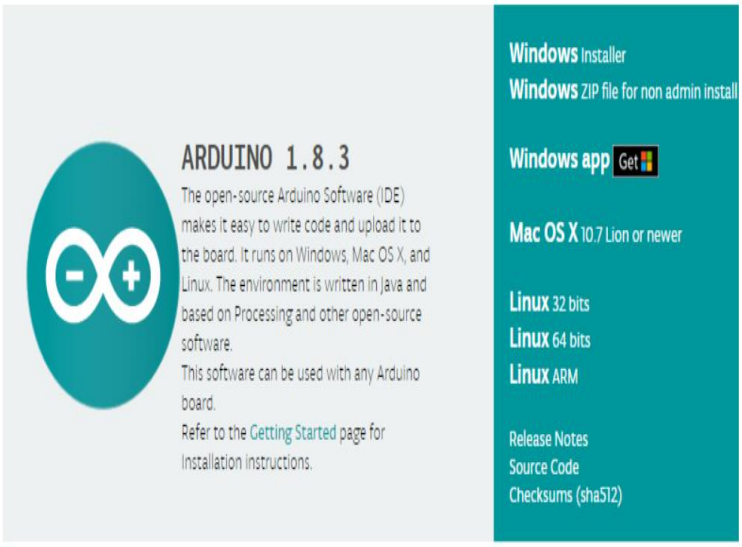
Figura 07: Imagem do Braço Robótico em funcionamento



Fonte: Manual de montagem da Robocore

4. INSTALAÇÃO DOS DRIVES E PROGRAMAS COMPATÍVEIS E NECESSÁRIOS

1° Passo: Instalando o Arduino IDE mais atual disponível



Fonte: *www.Arduino.cc*

O arduino IDE é um software, onde o sistema de código fonte é aberto, ou seja, está disponível para usar e fazer adaptações para cada projeto.

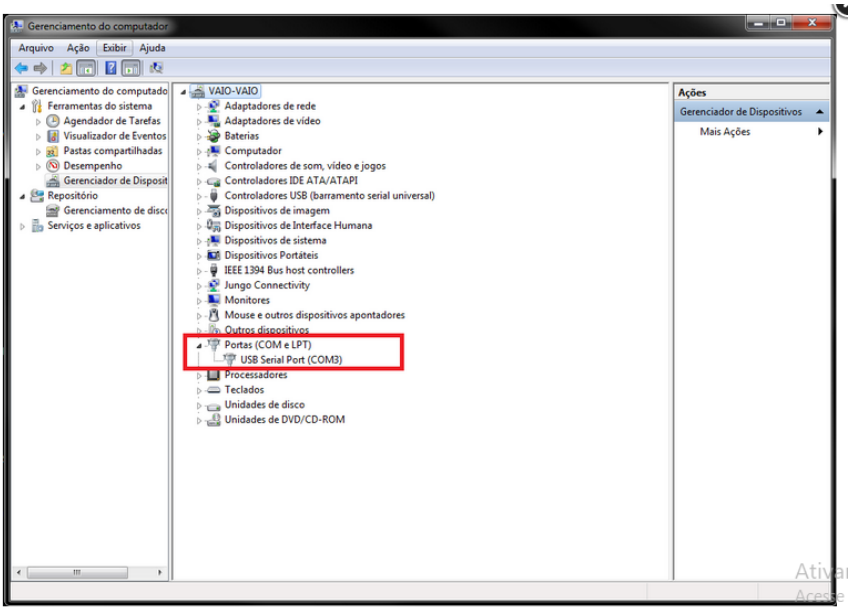
2º Passo: Instalando o drive Julieta

Para utilização da placa Julieta é necessário a instalação de um drive, que pode ser acessado no link: **Driver FTDI**, e em seguida realizar a escolha da versão do sistema operacional compatível.

		Processor Architecture							
Operating System	Release Date	x86 (32-bit)	x64 (64-bit)	PPC	ARM	MIPSII	MIPSIV	SH4	Comments
Windows*	2016-01-18	2.12.12	2.12.12	-	-	-	-	-	2.12.12 WHQL Certified Available as a setup executable Release Notes
Linux	2009-05-14	1.5.0	1.5.0	-	-	-	-	-	All FTDI devices now supported in Ubuntu 11.10, kernel 3.0.0-19 Refer to TN-101 if you need a custom VCP VID/PID in Linux
Mac OS X 10.3 to 10.8	2012-08-10	2.2.18	2.2.18	2.2.18	-	-	-	-	Refer to TN-105 if you need a custom VCP VID/PID in MAC OS
Mac OS X 10.9 and above	2015-04-15	-	2.3	-	-	-	-	-	This driver is signed by Apple
Windows CE 4.2-5.2**	2012-01-06	1.1.0.20	-	-	1.1.0.20	1.1.0.10	1.1.0.10	1.1.0.10	
Windows CE 6.0/7.0	2012-01-06	1.1.0.20 CE 6.0 CAT CE 7.0 CAT	-	-	1.1.0.20 CE 6.0 CAT CE 7.0 CAT	1.1.0.10	1.1.0.10	1.1.0.10	For use of the CAT files supplied for ARM and x86 builds refer to AN_319
Windows CE 2013	2015-03-06	BETA			BETA				BETA VCP Driver Support for WinCE2013

Fonte: Retirada do site https://www.robocore.net/tutoriais/kit-iniciante-robotica-eletronica

Com o driver instalado, conecte a placa ao pc via porta USB e em gerenciador de dispositivo, verifique qual a porta foi criada, conforme a imagem abaixo:

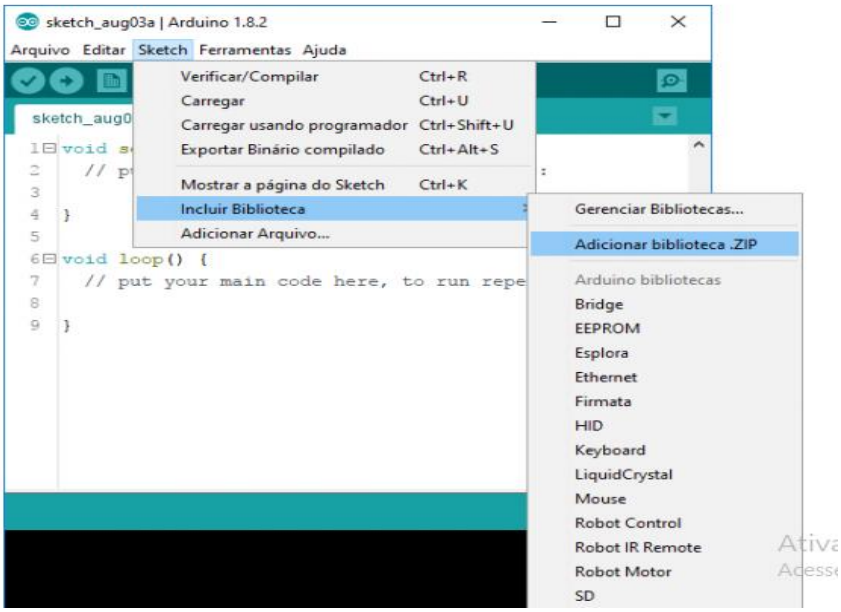


Fonte: Retirada do site https://www.robocore.net/tutoriais/kit-iniciante-robotica-eletronica

Na imagem, a porta criada foi a COM3, isso pode ocorrer ou diminuir/aumentar a numeração. Em seguida, é só selecionar a placa Arduino UNO, a porta e a interface do IDE estará pronta para uso.

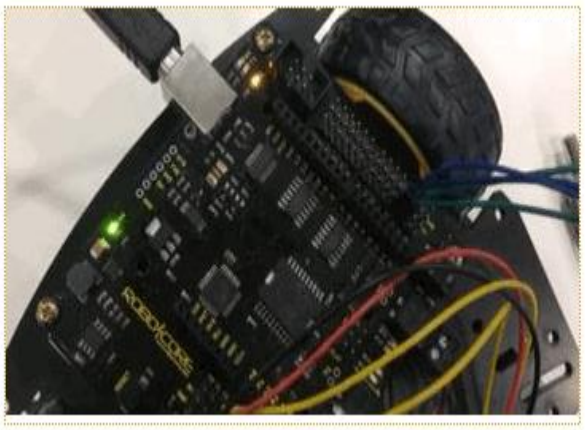
3° Passo: Instalando a biblioteca FalconRobot e os códigos das placas

Através do link: *Biblioteca FalconRobot*, *deverá ser feito o download do arquivo. Na sequência salve o arquivo no mesmo disco do IDE, abrá o menu da interface IDE e procure a opção Sketch->Incluir Biblioteca->Adicionar Biblioteca .ZIP, o arquivo baixado deve aparecer e deverá ser selecionado, como mostra a imagem abaixo.*



Fonte: Retirada do site https://www.robocore.net/tutoriais/kit-iniciante-robotica-eletronica

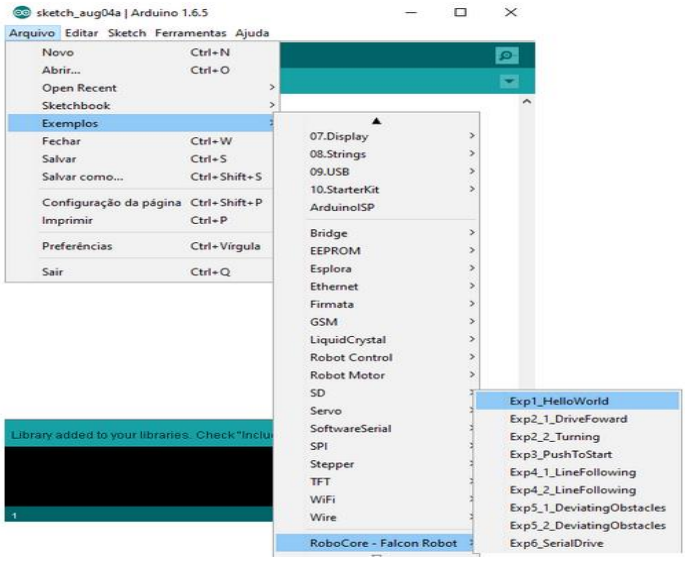
A placa está instalada e o seu robô pode receber os códigos de funcionamentos. Porém, é imprescindível que o primeiro teste com o mundo da programação seja feito, mesmo que esse seja bem simples. Então ligue o cabo USB AB e siga as próximas orientações.



Fonte: Retirada do site https://www.robocore.net/tutoriais/kit-iniciante-robotica-eletronica

- Inserindo os códigos na placa Julieta

Antes vamos testar a placa abrindo o IDE do Arduino e executando o Exp1_HelloWorld da seguinte forma: Arquivo->Exemplos->RoboCore Falcon Robot->Exp1_HelloWorld. Observe a imagem:



Fonte: Retirada do site https://www.robocore.net/tutoriais/kit-iniciante-robotica-eletronica

Ao executar esse exemplo, será possível observar o led da placa piscando em ciclos de 2 vezes a cada 1 segundo. A placa está respondendo em relação a programação inserida.

✓ Código para seguidor de linha

```
/*************************************************
*********************************
* Exp4_2_LineFollowing -- Falcon Robot Experiment 4.2
*
* This code reads the two line following sensors on A2 and
A3
* and prints them out to the Serial Monitor. Upload this
example to your
* Falcon Robot and open up the Serial Monitor by clicking
the magnifying glass
* in the upper-right hand corner.
*
* This is a real simple example of a line following algorithm.
It has
* a lot of room for improvement, but works fairly well for a
curved track.
* It does not handle right angles reliably -- maybe you can
come up with a
* better solution?
*
* Hardware setup:
* The Line Sensors must be connected to pins A2 and A3
(left and right sensor,
* respectively). The motors must be connected, and the
board must be receiving
* power from the battery pack.
*
* This sketch was written by RoboCore, with lots of help
from the Arduino
* community(especially from Sparkfun). This code is
completely free for any use.
*
*  Visit https://www.robocore.net/tutoriais/kit-iniciante-
robotica-introducao
* for Falcon Robot Kit Information
*
* 20 Jul 2017 MarceloFariaz
*************************************************
*********************************/
#include "FalconRobot.h"
// initialize a sensor object on A2 and A3
FalconRobotLineSensor left(A2);
FalconRobotLineSensor right(A3);
int leftValue;  // variable to store the left sensor value
int rightValue;  // variable to store the right sensor value
// constants that are used in the code. LINETHRESHOLD is
the level to detect
// if the sensor is on the line or not. If the sensor value is
greater than this
// the sensor is above a DARK line.
//
// SPEED sets the nominal speed
#define LINETHRESHOLD 700
#define SPEED 50  // Set to any number from 0 - 100.
FalconRobotMotors motors(5, 7, 6, 8);
int leftSpeed;   // variable used to store the leftMotor speed
int rightSpeed;   // variable used to store the rightMotor
speed
void setup() {
  Serial.begin(9600);
  Serial.println("Welcome to experiment 5.2 - Line
Following");
  delay(2000);
  Serial.println("Line Sensor Readings: ");
  delay(500);
}
void loop() {
  // Read the sensors
  leftValue = left.read();
  rightValue = right.read();
  // Print the sensors values
  Serial.print(leftValue);
  Serial.print("\t");  // tab character
  Serial.print(rightValue);
```

```
 Serial.println();   // new line
 // if the both sensors are on the line, drive forward left and
right at the same speed
 if((leftValue > LINETHRESHOLD) && (rightValue >
LINETHRESHOLD)) {
      leftSpeed = SPEED;
      rightSpeed = SPEED;
 }
 // if the line only is under the right sensor, adjust relative
speeds to turn to the right
 else if(rightValue > LINETHRESHOLD) {
      leftSpeed = SPEED + 40;
      rightSpeed = SPEED - 40;
 }
 // if the line only is under the left sensor, adjust relative
speeds to turn to the left
 else if(leftValue > LINETHRESHOLD) {
      leftSpeed = SPEED - 40;
      rightSpeed = SPEED + 40;
 }
 // run motors given the control speeds above
 motors.leftDrive(leftSpeed, FORWARD);
 motors.rightDrive(rightSpeed, FORWARD);
 delay(0);  // add a delay to decrease sensitivity.
}
```

- Inserindo os códigos na placa Arduino Sensor Shield V5

Como o objetivo é comandar os servos, será necessário programar a placa Arduino que está sendo utilizado. Para isso, basta conectar a placa ao computador, via cabo USB A/B e inserir o código abaixo:

```
**********************************************
***********/

//#define RESET_CONFIG

// -----------------------------------------------------------------
// -----------------------------------------------------------------
// Libraries

#include <EEPROM.h>
#include <Servo.h>

// -----------------------------------------------------------------
// -----------------------------------------------------------------
// Class - Joystick

class Joystick {
// ----------------------
public:
// ----------------------
// Constructor
// @params {pinX} : the pin for the X axis [int]
// {pinY} : the pin for the Y axis [int]
// {pinBtn} : the pin for the button [int] (default -1)
Joystick(int pinX, int pinY, int pinBtn = -1){
_pinX = pinX;
_pinY = pinY;
_pinBtn = pinBtn;
pinMode(_pinX, INPUT);
pinMode(_pinY, INPUT);
if(_pinBtn > -1)
pinMode(_pinBtn, INPUT_PULLUP);

_meanX = 500; // default
_meanY = 500; // default
}

// ----------------------
// Calibrate the mean axis of both axis
// @params {stream} : the stream to output the debug
messages [Stream*] (default NULL)
void calibrate(Stream *stream = NULL){
if(stream != NULL)
stream->print(F("Calibrating axis..."));

// calibrate
_meanX = 0; // reset
_meanY = 0; // reset
const int iterations = 10;
for(int i=0 ; i < iterations ; i++){
_meanX += getX();
_meanY += getY();
}
_meanX /= iterations;
_meanY /= iterations;

if(stream != NULL)
stream->println(F(" done!"));
}

// ----------------------
// Get the value of the X axis
// @returns the X axis value [int]
int getX(void){
return get(&_pinX);
}

// ----------------------
// Get the value of the Y axis
// @returns the Y axis value [int]
int getY(void){
return get(&_pinY);
```

```
}
// ----------------------
// Get the value of the button
// @returns the button value [int]
int getButton(void){
return get(&_pinBtn);
}
// ----------------------
// Get the mean value of the X axis
// @returns the X axis mean value [int]
int meanX(void){
return _meanX;
}
// ----------------------
// Get the mean value of the Y axis
// @returns the Y axis mean value [int]
int meanY(void){
return _meanY;
}
// ----------------------
// Print method overriden
size_t printTo(Print& p) const {
size_t size = 0;
size += p.print(F("x{ "));
size += p.print(getX());
size += p.print(F(" ; "));
size += p.print(meanX());
size += p.print(F(" } y{ "));
size += p.print(getY());
size += p.print(F(" ; "));
size += p.print(meanY());
if(_pinBtn > -1){
size += p.print(F(" } b{ "));
size += p.print(getButton());
}
size += p.println(F(" }"));
return size;
}
// ----------------------
private:
int _pinX, _pinY, _pinBtn;
int _meanX, _meanY;
// ----------------------
// Get the value of a pin
// @params {pin*} : the pointer to the pin [int*]
// @returns [int]
// > [0 - 1023] for an analog pin
// > [HIGH / LOW] for a digital pin
int get(int *pin){
if(pin == &_pinBtn){
return digitalRead(*pin);
} else {
analogRead(*pin); // firt call to change the channel
delay(5); // pause to charge the sample & hold circuit
return analogRead(*pin);
}
}
};
// ------------------------------------------------------------------
// Class - RoboServo
#define ROBOSERVO_MIN 0
#define ROBOSERVO_MAX 1
class RoboServo : public Printable, public Servo {
// ----------------------
public:
// ----------------------
// Constructor
RoboServo(int pin) : Servo() {
_limits[ROBOSERVO_MIN] = 0; // default - MIN
_limits[ROBOSERVO_MAX] = 180; // default - MAX
_pin = pin;
pinMode(pin, INPUT); // leave floating until attached
}
// ----------------------
// Attach the servo to the predefined pin
// @returns [uint8_t]
// > 0 if failure
// > the channel number
uint8_t attach(void){
Servo::attach(_pin); // attach the pin
int mean = _limits[ROBOSERVO_MIN] +
_limits[ROBOSERVO_MAX];
mean /= 2;
Servo::write(mean); // default angle
}
// ----------------------
// Get the MAX limit
// @returns [int]
int getMAX(void){
return getLimit(ROBOSERVO_MAX);
}
// ----------------------
// Get the MIN limit
// @returns [int]
int getMIN(void){
return getLimit(ROBOSERVO_MIN);
}
// ----------------------
// Print method overriden
size_t printTo(Print& p) const {
size_t size = 0;
size += p.print(Servo::attached());
size += p.print(F(" { "));
size += p.print(_limits[ROBOSERVO_MIN]);
size += p.print(F(" ; "));
size += p.print(Servo::read());
```

```
size += p.print(F(" ; "));
size += p.print(_limits[ROBOSERVO_MAX]);
size += p.println(F(" }"));
return size;
}
// ----------------------
```

```
// Set the MAX limit
// @params {value} : the maximum angle [int]
void setMAX(int value){
setLimit(ROBOSERVO_MAX, value);
}
// ----------------------
```

Como esse código é muito extenso, foi adicionado um apêndice 05 com o arquivo completo.

5. FUNCIONAMENTO DO KIT ROBÓTICA

5.1 Robô Falcon: Este modelo de Robô possibilitará ao aluno ter contato com a programação de uma placa de arduino e realizar movimentos em um carro que obedecerá a comandos específicos, como andar em linha reta, simulando o movimento retilíneo e assim poder realizar uma análise precisa sobre aceleração, velocidade, energia e trabalho da força resultante. Para tal funcionamento, após a montagem e instalação dos códigos apropriados e ajustados ao que se pretende, os percursos podem ser definidos pelos alunos, pois o robô é um seguidor de linha preciso.

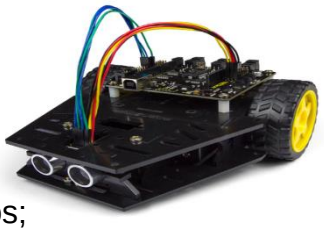
Para tal procedimento ocorrer, o ambiente é bem simples e fácil de montar. Os alunos irão construir junto com o professor uma pista, seja em linha reta ou curvilínea, utilizando uma fita isolante preta, afim de criar a interação entre o sensor de refletância e a superfície escura. O carro Robô irá procurar, pela programação utilizada neste manual, a fita preta a todo instante, realizando um movimento para análise dos conceitos físicos de cinemática e dinâmica.

5.2 Braço Robótico: A interação entre um equipamento robótico e outros materiais, geram nos autores do experimento uma maior sensibilidade aos conceitos físicos inseridos, dessa forma, o braço robótico é um equipamento que atende essa necessidade do educando. Este kit de robótica age de forma autônoma, a partir da montagem e instalação do código utilizado neste manual, o aluno poderá realizar movimentos, simulando um braço, afim de levantar e descer objetos para observar tempo, altura, peso, energia potencial gravitacional e potência do equipamento.

6. APLICAÇÃO DAS ETAPAS E TRATAMENTO DE DADOS

Roteiro 01

Aplicando o Teorema da Energia Cinética



Objetivos:

Analisar a variação da velocidade do robô falcon;

Medir os tempos em pontos do espaço determinados;

Aplicar o Teorema da Energia Cinética, no movimento realizado pelo falcon.

Materiais:

Papel 40;	Fita métrica;
Fita isolante;	Pilhas e carregadores;
Cronômetros;	Robô Falcon montado.

Sequência de Procedimentos

a) Fixar 2 metros de papel branco (cartolina ou papel 40), em uma bancada, utilizando para isso, cola quente ou fita durex;
b) Fixar sobre o papel, uma fita isolante preta, formando um trajeto reto, com o mesmo comprimento de 2 metros;
c) Realizar marcações de posições S_1 = 40 cm, S_2 = 80 cm, S_3 = 120 cm e S_4 = 160 cm.
d) Colocar o carro programado (Robô Falcon) em movimento na trajetória retilínea;
e) Através de cronômetros anotar os tempos gastos para o carro atingir cada posição marcada;
f) Aferir a massa do carro;
g) Repetir o procedimento três vezes, afim de minimizar os erros de contagem;
h) Realizar o preenchimento de uma tabela, para facilitar os cálculos de Energia e Trabalho Mecânico;
i) Efetuar os cálculos e responder os questionamentos do experimento.

Tabela para preenchimento de dados

Tabela: Posições em centímetros e tempos em segundos

	POSIÇÕES (cm)			
TEMPO (s)	**40 cm**	**80 cm**	**120 cm**	**160 cm**
$\mathbf{t_1}$				
$\mathbf{t_2}$				
$\mathbf{t_3}$				
$\bar{t}$				

Fonte: Autor

Sugestão para cálculos

a) O aluno deverá considerar a velocidade inicial do carro igual a zero;
b) Realizar as transformações de unidades, afim de obter os resultados no Sistema Internacional (SI);
c) Visualizar a partir de qual posição o movimento tende a ser com velocidade constante;
d) Definir a velocidade média no percurso, onde a velocidade variou;
e) Utilizar a equação de Energia Cinética:
    a. $Ec = \frac{1}{2} . mv^2$
f) Aplicar o Teorema da Energia Cinética:
    a. $w = \frac{1}{2} . mv_o^2 - \frac{1}{2} . mv^2$

Questionamento e Discursões

a) O carro utilizado no experimento é programável, cite quais os requisitos para a montagem de comando do Robô Falcon!
b) Com a coleta de dados foi possível visualizar onde o carro passou a não mais ser acelerado, explique quais os fatores podem ter influenciado nessa alteração!
c) No último trecho realizado pelo Robô Falcon não podemos aplicar o Teorema da Energia Cinética. Explique quais os motivos!
d) Seria possível aplicar o Teorema da Energia Cinética para obter, no experimento em questão, a Força Resultante o trecho onde a velocidade sofreu alterações?

Roteiro 02

Calculando a Potência do Braço Robótico

Objetivos:

Medir o tempo gasto pelo Braço robótico para o levantamento de diferentes massas;

Calcular a Energia Potencial gravitacional, para cada massa utilizada no experimento;

Aplicar a equação de potência, afim de quantificar e caracterizar os servos do Braço robótico.

Materiais:

03 x Massas de cobre com valores diferentes;
Balança analógica de precisão;
Cronômetros;
Fita métrica;
Pilhas e carregadores;
Braço Robótico montado

Método para realizar o procedimento

a) Medir as massas disponíveis, utilizando uma balança;
b) Usar a trena para medir as alturas máximas atingidas pelo Braço Robótico;
c) Levantar uma massa m, até atingir a altura máxima e repetir essa ação até três vezes;
d) Outro aluno irá cronometrar o tempo gasto;
e) Anotar as informações para organizar tais informações em forma de tabela.
f) Realizar o cálculo da Energia Potencial Gravitacional para cada massa do experimento;
g) Calcular a Potência, utilizando os dados coletados.
    a. $P = \frac{Epg}{\bar{t}}$, onde Epg é Energia Potencial Gravitacional e $\bar{t}$ será a média dos tempos obtidos para cada massa.

h) Converter as unidades para o Sistema Internacional (S.I)

Sugestão para anotações de dados

Solicitar que os alunos de cada grupo, anotem os dados em uma tabela, transforme as unidades de medida para o Sistema Internacional de Unidades e em seguida, realizem o cálculo da Potência do Braço Robótico.

Tabela: Tempos em segundo e massas em grama.

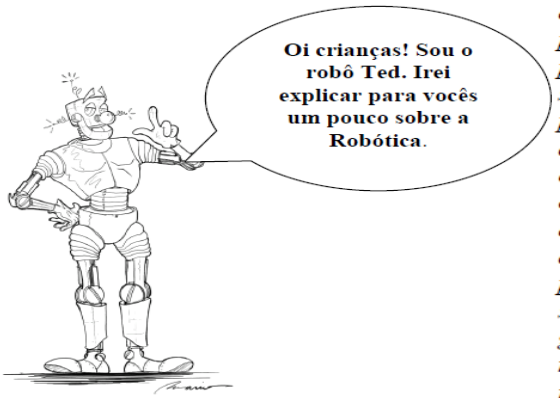
	Tempo (s)			**Média dos tempos**	**Potência (w)**
Massa (g)	t_1	t_2	t_3	$\bar{t}$	**P**
m_1=					
m_2=					
m_3=					

Fonte: Autor

Questionamento e Discursões

a) Quais fatores podem influenciar na diferença, caso exista, dos tempos obtidos para uma mesma massa, durante o levantamento feito pelo Braço Robótico?
b) Como se trata de um mesmo equipamento (Braço Robótico) nos procedimentos de coleta de dados e preenchimento da tabela, discuta os resultados para a potência?
c) As massas que serão erguidas podem possuir Energia? Podem possuir Trabalho? Faça a diferença, caso necessário.
d) Qual a relação entre Trabalho da força peso e Potência do Braço Robótico.

7. SUGESTÃO DE ATIVIDADES PARA APLICAÇÃO APÓS APLICAÇÃO DOS ROTEIROS.



Na Robótica, aprendemos a montar, controlar e programar robôs. Mas, para isso, precisamos do auxilio de diversas peças, que podem estar presentes em kits de montagem ou serem construídas por alguém. Cada peça possui sua própria função, elas nos ajudam a dar forma ao robô, como também faz com que ele se movimente e realize as atividades que ordenamos. As peças são conectadas a um computador que pode ser chamado de cérebro do robô. Exemplos de peças podem ser rodas parecidas as dos carros, porém menores, sensores que servem para sentir o ambiente, garras que servem para segurar coisas e os motores que fazem com que as partes do robô se mexam.

Questão 01. O que é o que é?

Para eu funcionar me ligam ao cérebro do robô, mas não sou um sensor. Funciono com eletricidade, mas não sou um rádio. Se quiser que alguma parte do robô se mexa, vai precisar de mim. Quem eu sou?

a) Motor
b) Sensor
c) Garra
d) Parafuso

Questão 02. O que é o que é?

Eu poso sentir as coisas, mas não sou nem mão, nem língua, nem nariz, nem olhos e nem ouvido. Para eu ser usado preciso me conectar ao cérebro do robô, mas não sou um motor. Quando eu não estou em um robô, ele fica perdido e precisa que alguém lhe diga o que está acontecendo ao seu redor. Quem sou eu?

a) Motor
b) Sensor
c) Garra
d) Parafuso

Questão 03. Um ciclista desce uma rua inclinada, com forte vento contrário ao seu movimento, com velocidade constante. Pode-se afirmar que:

a) sua energia cinética está aumentando.
b) sua energia potencial gravitacional está diminuindo
c) sua energia cinética está diminuindo.

d) sua energia potencial gravitacional é constante.
e) não há variação na energia potencial gravitacional.

Questão 04. Baseado nos conceitos apresentados durante essas oficinas de Física, redija um texto, mostrando o que você entendeu sobre energia?

Resp.:_____
_____
_____
_____
_____
_____
_____
_____

Série:_____

Turma:_____**Turno:**_____

Aluno(a):_____

Data:_____**/**_____**/**_____