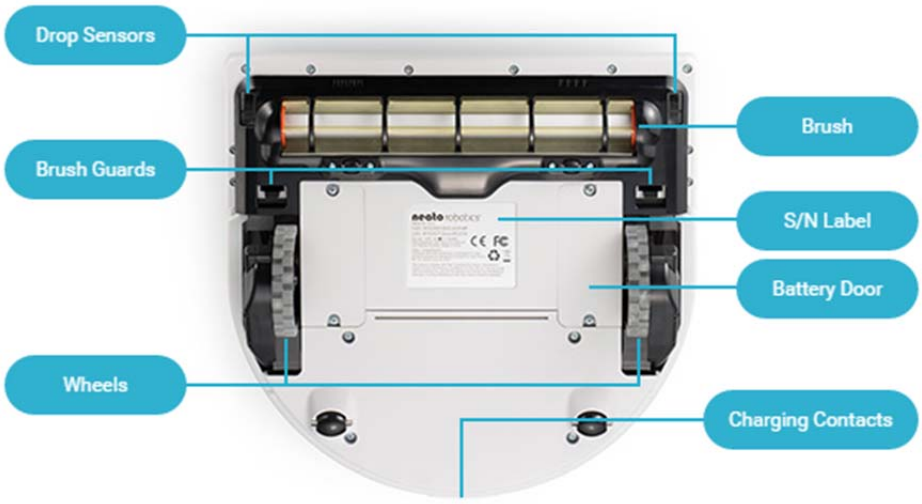
Robot & User Interface description

1. Robot description

It's a commercial model from the brand Neato, specifically the model XV-Essential.

What sets it apart from the other brands like 'Roomba' is its 360 degrees rotatory laser sensor which is interesting for SLAM projects. It also has other sensors, find them described below:

- Single wall sensor on the front right side.
- 4 bumpers. Two in the front, one in each side.
- 2 fall sensors, on the front bottom, on the sides.
- 360 degrees rotatory laser
- Wheel encoders with 1mm resolution.
- Accelerometer



2. Robot measures

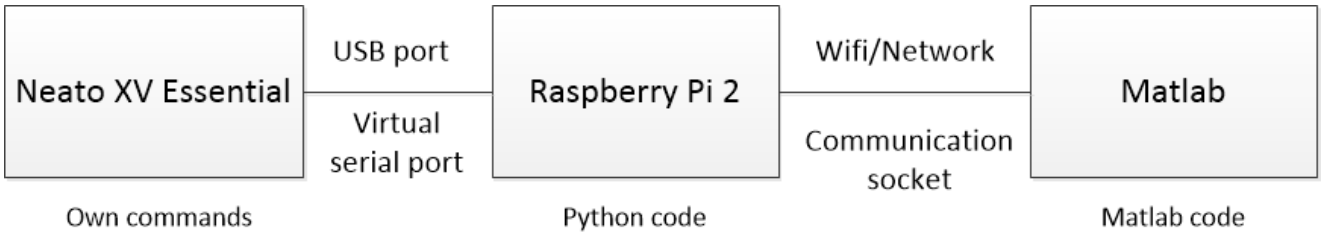
Find below two diagrams with the robot measures.

Wheel radius = 3.85 cm

3. Communication interface description

The robot has an USB port which acts as a serial communication port. This serial port allows us to send commands in order to get sensor data and to give movement orders to the robot.

We have used a Raspberry Pi 2 acting as a communication interface.
In one side, the Raspberry connects trough USB to the Neato, and in the other side, it has a Wifi dongle publishing a communication socket. The use of this socket allows us to connect to the Raspberry and controlling the Neato using its own commands.



Commands accepted by NEATO robot are listed at "programmersmanual_20140305.pdf".

DEMO CODE (in MATLAB)

```
function demo()

%######## Global vars ######
hist = {}; % Will record all the robot data

%######## Code ##########
% UI Neato connection info
prompt = {'Connection information',};
dlg_title = 'Connection';
num_lines = 1;
def = {'192.168.1.12:20000',};
conn = inputdlg(prompt,dlg_title,num_lines,def);
conexion = strsplit(conn{1},':');

% Neato Connection
sck = socket(conexion{1},str2double(conexion{2}));
rs = sck.connect(sck);

% Wait lidar start
wait_lidar();

%### Demo robot movement and data acquisition ###

%Read robot data & log it into "hist"
msg = ['GetMotors LeftWheel RightWheel' , char(10)];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,1} = data;
msg = ['GetLDSScan' , char(10)];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,2} = data;

% SetMotor distance in [mm], speed in [mm/s]
% Direct movement
dist = 1000; %[mm]
speed = 120; %[mm/s]
msg = ['SetMotor LWheelDist ', num2str(dist) ,' RWheelDist ', num2str(dist)
, ' Speed ', num2str(speed), char(10)];
sck.sendMsg(sck,msg);

%Wait until movement stops
pause(dist/speed);

%Read robot data & log it into "hist"
msg = ['GetMotors LeftWheel RightWheel' , char(10)];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,3} = data;
```

```
msg = ['GetLDSScan' , char(10)];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,4} = data;

% 90º rigth turn
dist = 190;
speed = 120;
msg = ['SetMotor LWheelDist ', num2str(dist) ,' RWheelDist ', num2str(-
dist) , ' Speed ', num2str(speed), char(10)];
sck.sendMsg(sck,msg);

% Wait until movement stops
pause(dist/speed);

%Read robot data & log it into "hist"
msg = ['GetMotors LeftWheel RightWheel' , char(10)];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,5} = data;
msg = ['GetLDSScan' , char(10)];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,6} = data;

% Direct movement
dist = 1000; %[mm]
speed = 120; %[mm/s]
msg = ['SetMotor LWheelDist ', num2str(dist) ,' RWheelDist ', num2str(dist)
, ' Speed ', num2str(speed), char(10)];
sck.sendMsg(sck,msg);

%Wait until movement stops
pause(dist/speed);

%Read robot data & log it into "hist"
msg = ['GetMotors LeftWheel RightWheel' , char(10)];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,7} = data;
msg = ['GetLDSScan' , char(10)];
data = sck.sendMsg(sck,msg);
display(data);
hist{1,8} = data;

% Save data into workspace
assignin('base', 'hist', hist);

% Close connection
sck.sendClose(sck);

%######## End Code ##########

%######## Functions ##########
function wait_lidar()
wt = waitbar(0,'Waiting Lidar Start...');
for p=1:4
    pause(1);
    waitbar(p/5,wt,'Waiting Lidar Start......');
end
close(wt);

end

end
```

```
%######## Socket functions ##########


classdef socket
    properties
        skt
    end
    methods(Static)
        %Contructora
        function obj = socket(ip,port)
            if nargin > 0
                obj.skt = tcpip(ip,port,'NetworkRole','Client');
                set(obj.skt, 'BytesAvailableFcnMode', 'terminator');
                set(obj.skt, 'InputBufferSize', 12288);
                set(obj.skt, 'terminator', 10);

                %obj.skt = tcpip(ip,port,'NetworkRole','Client');
                %set(obj.skt, 'BytesAvailableFcnMode', 'packet');
                %set(obj.skt, 'InputBufferSize', 12288);
                %set(obj.skt, 'terminator', 26);

                %obj.EOSMode = 'read';
                %obj.EOSCharCode = 169;
            end
        end

        %Conecta con el socket en la ip y puerto indicados
        function stat = connect(obj) %127.0.0.1, 20000
            fopen(obj.skt);
            stat = fscanf(obj.skt); % Recibe mensaje de bienvenida
        end

        %Obtiene mensaje
        function txt = getMsg(obj)
            txt = fscanf(obj.skt);
        end

        %Envia mensaje y espera respuesta
        function txt = sendMsg(obj,msg)
            fprintf(obj.skt,msg);
            txt = fscanf(obj.skt);
        end

        function sendClose(obj)
            fprintf(obj.skt,'Close');
        end

        %Cierra el socket
        function close(obj)
            fclose(obj.skt);
            %delete(sck);
            %clear sck;
        end
    end
end

% Se pueden definir otras funciones para usarlas dentro de la clase
%http://es.mathworks.com/videos/developing-classes-overview-68982.html
```