LINE FOLLOWING BEHAVIOR FOR AN AUTONOMOUS MOBILE ROBOT USING ARTIFICIAL NEURAL NETWORKS

Manash Kumar Mandal
1203043
Department of Electrical & Electronic Engineering
Khulna University of Engineering & Technology

Contents

Abstract

In order to achieve tasks by the mobile robots, these robotic systems must have been intelligent and should decide their own action. To guarantee the autonomy and the intelligence for line following behavior, it is necessary to use the techniques of artificial intelligence like the artificial neural networks. This project report presents an approach for line following task by an autonomous mobile robot using a single layer neural network. The proposed controller is used for following any line on a plain surface with different width. This controller can also be upgraded to determine the value of k_p and k_d and make the autonomous line following a PD controller based. The results acquired from Neural Network simulation and implementation on the robot are shown and discussed.

Keywords- Feedforward Neural Network, Machine Learning, Robotics, Backpropagation Algorithm, Stochastic Gradient Descent, Proportional Derivative Controller

1 Introduction

The line follower is a self operating robot that detects and follows a line that is drawn on the floor. The path consists of a black line on a white surface (or it may be reverse of that). The control system used must sense a line and maneuver the robot to stay on course, while constantly correcting the wrong moves using feedback mechanism, thus forming a simple yet effective closed loop system. The robot is designed to follow very tight curves.

In this project, the conventional control system is being replaced by artificial neural network.

1.1 Line detection mechanism

Line can be detected by either using Infra-red (IR) sensors, Light Dependent Resistor (LDR) or by a camera with line detection algorithms. Line tracking is a very important notion in the world of robotics as it give to the robot a precise, error-less and easy to implement navigation scheme.

Detecting line using IR To detect line using IR, a threshold value must be measured. IR sensor gives different reading on different colored surface. Both values from IR on white line and IR on black line must be recorded before sensing the line.

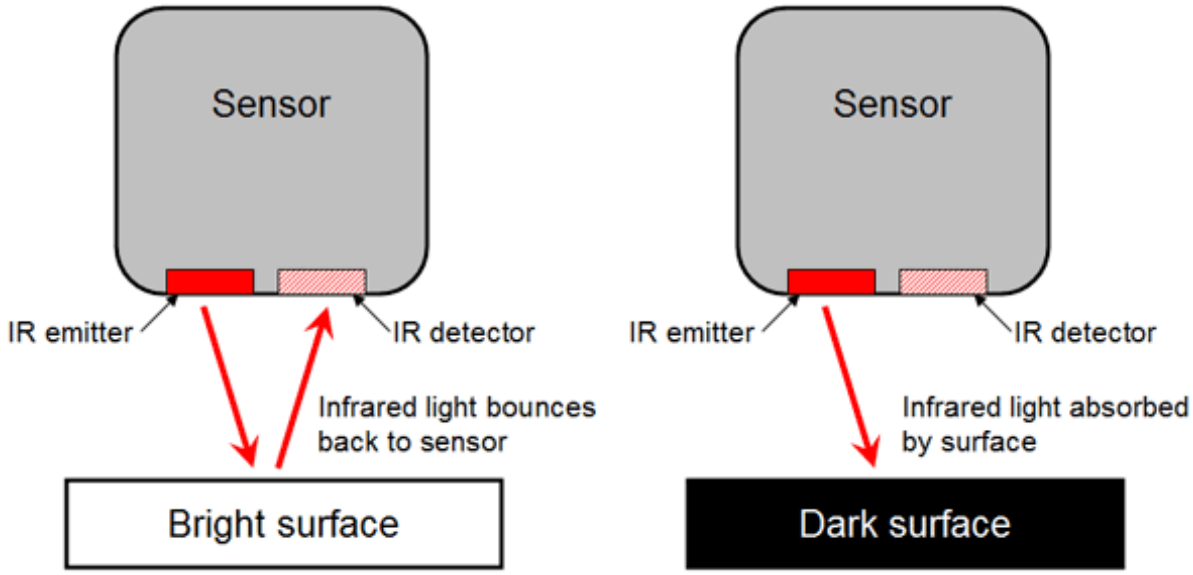

Figure 1: Detecting line using IR sensor

Detecting line using LDR To detect line using LDR same procedure from IR can be used. Since reflected line intensity depends on the reflecting surface. If the color of the surface is white, maximum light is reflected. If it is black then minimum light is reflected. Reflected light has different intensity based on the color of the surface it is being reflected from. So, nearest colors can be differentiated using LDR sensors.

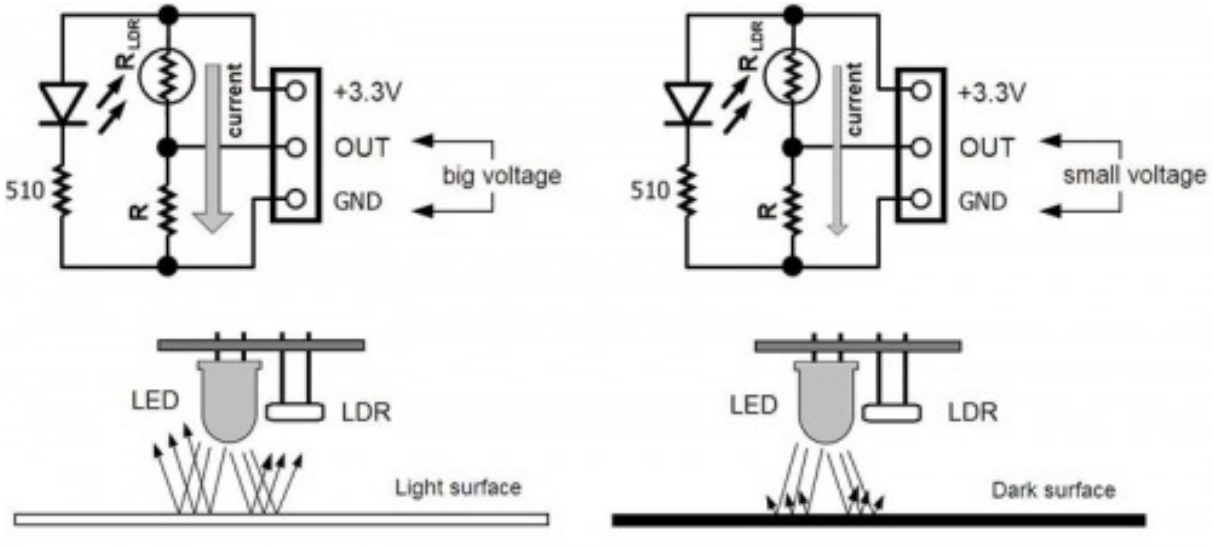

Figure 2: Detecting line using LDR sensor

1.2 Algorithm for detecting line

Algorithm 1 Line Detecting Algorithm

```
1:  procedure DETECTLINE(irPin)
2:      irPin ← ir receiver pin
3:      irReading ← analog reading from ir receiver
4:      threshold ← threshold value for differentiate betweeen white and black line
5:      irDigitalReading ← converts analog into binary format
6:      irReading ← reading from irPin
7:      if irReading > threshold then
8:          irDigitalReading ← 1
9:      else
10:         irDigitalReading ← 0
```

1.3 Getting current position of the robot using IR / LDR Array

Current position of a robot on a line can be extracted by using IR array consisting of two or more than two IR/LDR sensors. Suppose, a robot has an IR array of 5 IR sensors. The spacing between two IR sensor is **1cm**, and the width of the line is **1.5cm**. At any time, when the robot is on the line, two values will differ from other three values meaning robot is facing straight, leaning left or leaning right. Depending on the position of the robot on the line, speed of motors can be varied to keep it on the line. This process is completely experimental and varies with the body, circuitry, battery rating, motor rating and mostly other things. The position value can be returned either in binary form such as **01100** using 1 or in weighted value such as **2500** from 2.

1.3.1 Algorithm for weighted positional value

Algorithm 2 Position calculating algorithm

```
1:  procedure CALCULATEPOSITION(numberOfSensors)
2:      numberOfSensors ← number of ir/ldr sensors
3:      numberOfActiveSensors ← 0
4:      digitalReading[numberOfSensors] be new array
5:      weightedValues ← 0
6:      weight ← 1000                          ▷ Setting weighted value 1000
7:      position ← −1                          ▷ Setting current position at -1
8:      for index ← 0 to numberOfSensors do
9:          digitalReading[index] ← DETECTLINE(index)
10:         if digitalReading[index] ← 1 then
11:             numberOfActiveSensors ← numberOfActiveSensors + 1
12:         weightedValues ← weightedValues + digitalReading[index] ∗
    weight
13:     position ← weightedValues/numberOfActiveSensors
14:     return position
```

1.4 Conventional line following mechanism

Conventional line following robots follow lines on a surface based on either predefined conditions or line patterns. Position of the robot can be calculated from 2. If the position of the sensor indicates the robot is being shifted to right from mid point of the line, then the speed of left motor is increased and right motor is decreased and vice versa. If the position of the robot is at middle point then both of the motor will go in the same direction with same speed. The procedure can be viewed from figure 3.

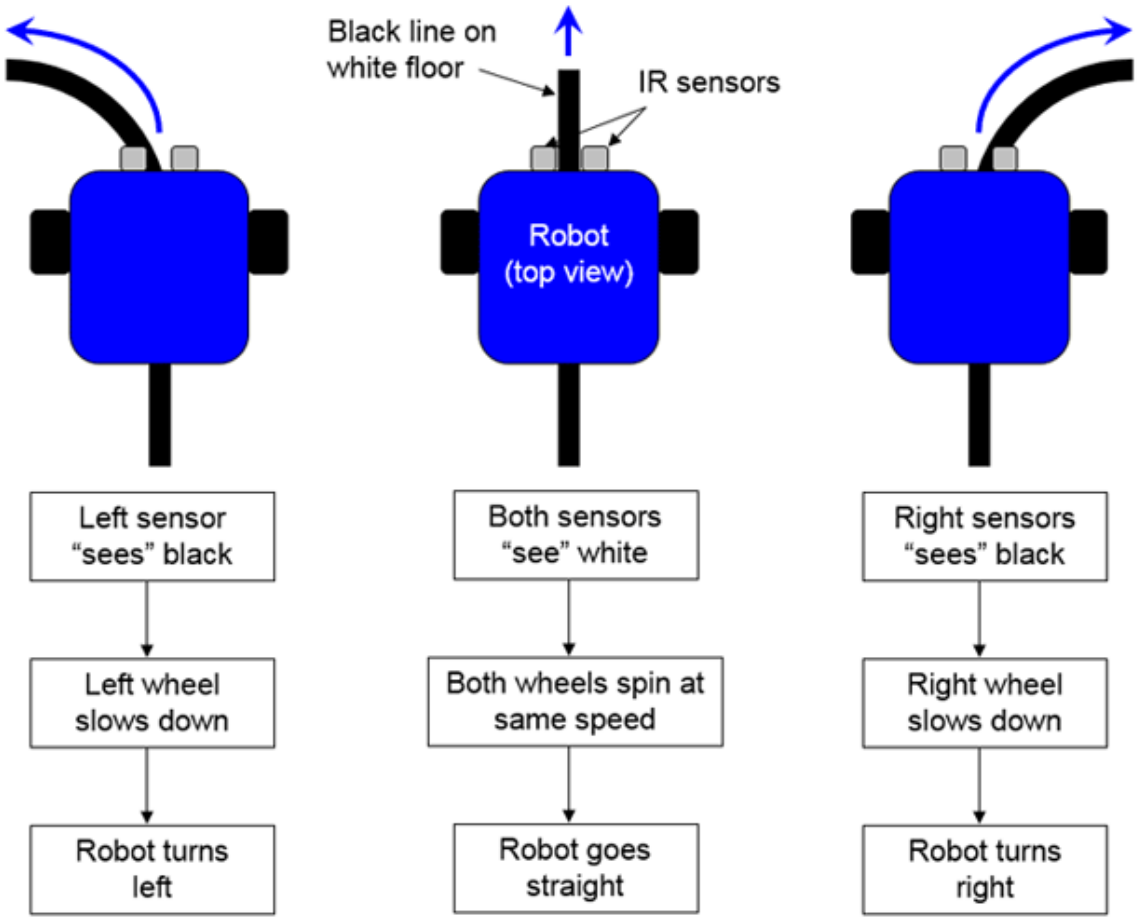

Figure 3: Differential steering drive for following line

1.5 Drawbacks of line following robot based on differential drive system

Some drawbacks of differential drive system

1. It is a static method that can only be used on a specific robot for which the algorithm was designed
2. Conditional statements varies with
   (a) Weight of the robot
   (b) Speed of the robot
   (c) Spacing between the IR/LDR sensors of the sensor array
   (d) Number of IR/LDR sensors used to make the array
   (e) Width of the line to be followed by the robot

3. This method is not appropriate for driving the robot in right and acute angle turns

2 Machine Learning

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed".

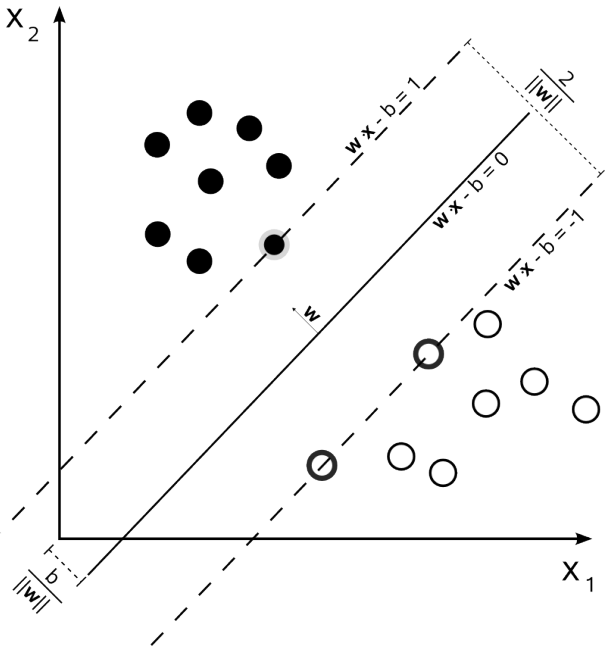

Figure 4: Support Vector Machine (SVM), an algorithm to classify data

2.1 Artificial Neural Network

In machine learning and cognitive science, artificial neural networks (ANNs) are a family of models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which exchange messages between each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

For example, a neural network for handwriting recognition is defined by a set of input neurons which may be activated by the pixels of an input image. After being weighted and transformed by a function (determined by the network's designer), the activations of these neurons are then passed on to other neurons. This process is repeated until finally, an output neuron is activated. This determines which character was read.

Like other machine learning methods systems that learn from data neural networks have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including computer vision and speech recognition.

A basic model of neural network is displayed in figure 5.

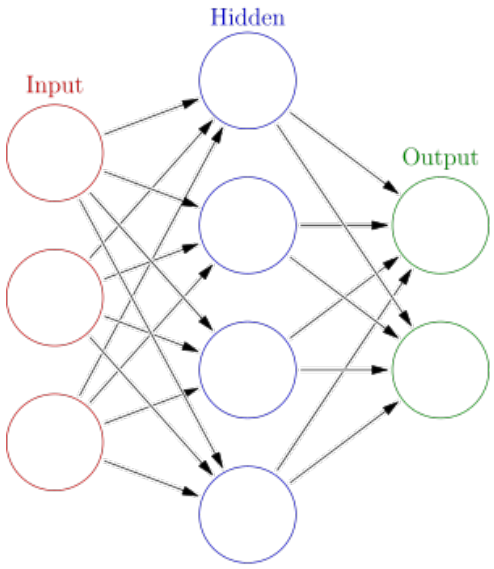

Figure 5: Basic model of artificial neural networks

2.2 Feedforward Neural Network

A feedforward neural network is an artificial neural network where connections between the units do not form a cycle. This is different from recurrent neural networks.

The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

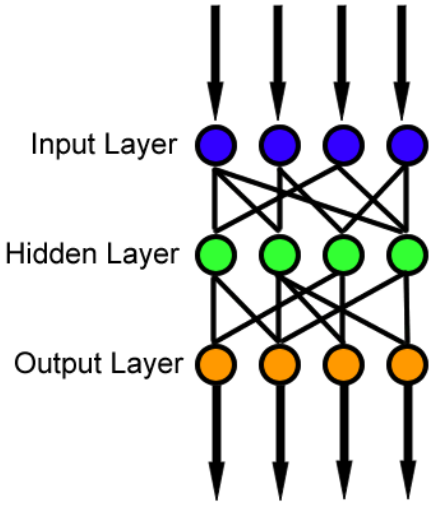

Figure 6: A simple Feedforward neural network.

2.2.1 Multi-layer perceptron

This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function.

The universal approximation theorem for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds for a wide range of activation functions, e.g. for the sigmoidal functions.

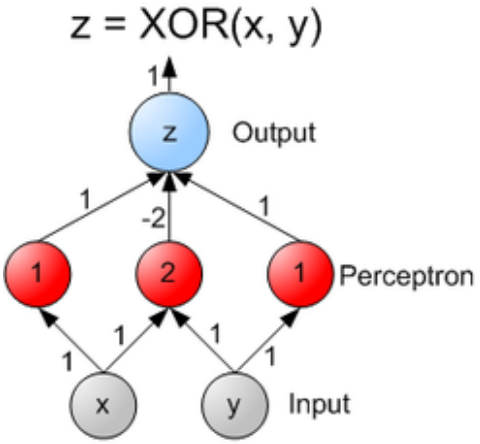

Figure 7: A multi-layer perceptron network learning XOR

2.2.2 Backpropagation

Backpropagation, an abbreviation for "backward propagation of errors", is a common method of training artificial neural networks used in conjunction with an optimization method such as gradient descent. The method calculates the gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

2.2.3 Gradient descent

Gradient descent is a first-order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point.

3 Importance of the project

Robotics technology is emerging at a rapid pace, offering new possibilities for automating tasks in many challenging applications, especially in autonomous self driving vehicles. A lot of parameters and conditions and other necessary things are needed to be considered to build a complete autonomous self driving vehicles, yet making the vehicle to learn to follow the line of the path is one of the basic building blocks to build a complete autonomous self driving vehicles. The main objective of the project is to train a network and apply it on a autonomous line following prototype which can navigate autonomously at any line consisting of any width.

Parameters of vehicles can be different from each other but the training techniques to follow a line on a surface will be the same for all of the test objects. So it is more convenient to train the network and set the weight than defining all the conditions to follow line.

4 Prototype construction

List and description of the items and electronic modules used in this project.

4.1 Arduino Nano

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x) or ATmega168 (Arduino Nano 2.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one. The Nano was designed and is being produced by Gravitech.

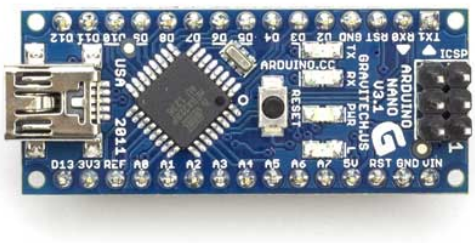

Figure 8: Arduino Nano Front View

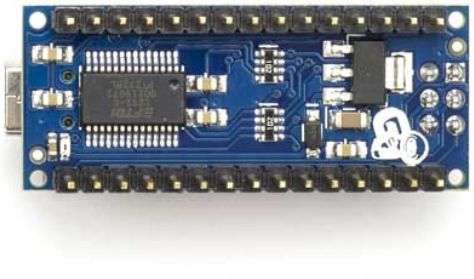

Figure 9: Arduino Nano Back View

4.1.1 Technical Specifications

Microcontroller	Atmel ATmega168 or ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz
Dimensions	0.73" x 1.70"
Length	45 mm
Width	18 mm
Weigth	5 g

4.2 QTR-8A Reflectance Sensor Array

This sensor module has 8 IR LED/phototransistor pairs mounted on a 0.375" pitch. Pairs of LEDs are arranged in series to halve current consumption, and a MOSFET allows the LEDs to be turned off for additional sensing or power-savings options. Each sensor provides a separate analog voltage output.

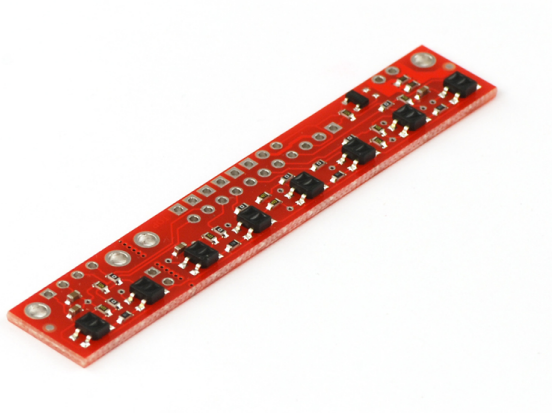

Figure 10: Pololu QTR-8A Reflectance Sensor Array

4.3 L298N Motor Driver

To drive two motors, L298N stepper motor driver breakout was used.

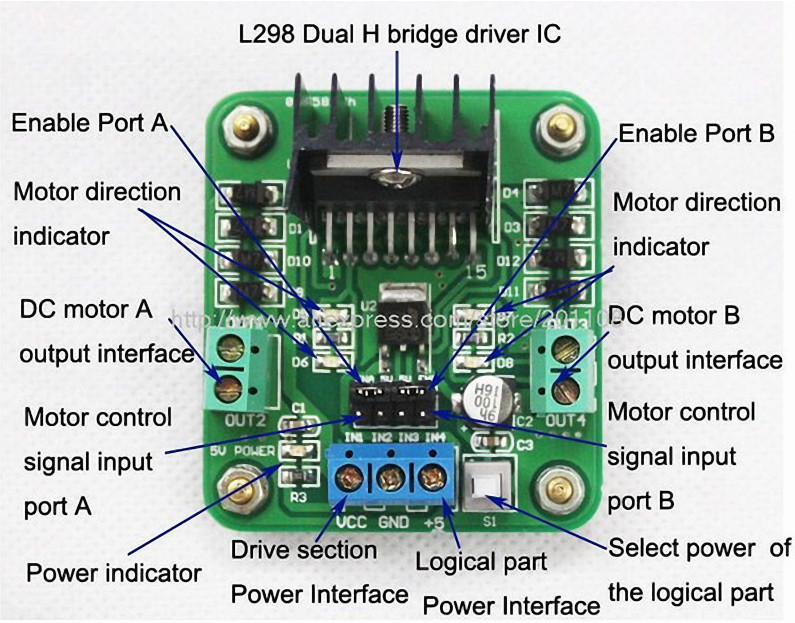

Figure 11: L298N DC Motor Driver

4.4 Lithium-Polymer Battery

A 3 cell Li-Po battery was used with 800mAh capacity.

Figure 12: 3 Cell 800mAh Li-Po Battery

4.5 Bluetooth Module

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth

system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as $12.7mm \times 27mm$.

Hardware Features

1. Typical $-80dBm$ sensitivity
2. Up to $+4dBm$ RF transmit power
3. Low Power 1.8V Operation
4. UART interface with programmable baud rate
5. With edge connector

Figure 13: HC-05 Bluetooth Module

4.6 Breadboard

For completing the circuit of the prototype I used small breadboard to connect the wires.

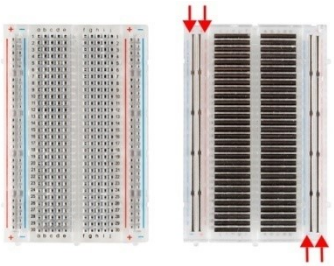

Figure 14: Half size breadboard (clear)

4.7 Premium Jumper Wire

I used male to male and male to female jumper connectors to connect the components together.

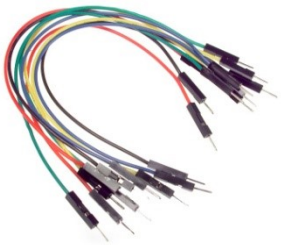

Figure 15: Connecting wires