On Your Track (O.Y.T) Robot

Rodrigo Moreno Garcia

Illinois Institute of Technology

3201 S State Street

Box #2225

Chicago, IL - 60616

+1-312-401-5853

rmorenog@hawk.iit.edu

Nicolas Ribeiro Vieira

Illinois Institute of Technology

3201 S State Street

Box #2799

Chicago, IL - 60616

+1 312-478-3684

nribeiro@hawk.iit.edu

Leonardo Pavanatto Soares

Illinois Institute of Technology

3241 S Wabash Ave

Box #639

Chicago, IL - 60616

+1-312-478-3803

ipavanat@hawk.iit.edu

Igor de Sousa Dantas

Illinois Institute of Technology

3201 S State Street

Box #2287

Chicago, IL - 60616

+1 312-478-3777

idesousa@hawk.iit.edu

Pedro Henrique de Brito Souza

Illinois Institute of Technology

3201 S State Street

Box #2218

Chicago, IL - 60616

+1-773-666-3970

pdebrito@hawk.iit.edu

ABSTRACT

This paper presents the development of a multi-task line follower robot. It will be referenced as O.Y.T. (On Your Track Robot). It comprehends the understanding of the sensors behaviors and functionality, PCB elaboration, software operations such as robot control. The robot is mainly composed of: infrared sensors to detect the color difference between the track and the surface, an ultrasonic sensor to be able to stop without hitting an obstacle, transistors to control both DC motors, LI-PO batteries and an XBee module to receive and send data to a stationary computer that has another XBee and the MATLAB software. The stationary computer uses MATLAB to control the robot tasks, as well as receive its status and the data collected by the sensors through the XBee.

Keywords

MATLAB, Arduino, robot, line-follower, data, collector, supplier, XBee.

# INTRODUCTION

Although line-follower robots are very present in many robot competitions around the world, the competition’s approach is purely limited to ride along the track in the shortest time. Therefore, the current type of line-followers has almost zero application.

Moreover, technology is a dominant force nowadays, and the more it penetrates in even more aspects of human’s life, the more autonomous and automated things become. Many different techniques are used to make technology usable and accessible to everyone, from engineers to end users, but such techniques come with the price of high amounts of investment to make it even more independent of human interaction.

With these thoughts in mind, it is proposed the development of a simple, user friendly and reliable device. It should be capable of performing automated actions with almost no necessity of human interaction, and using one of the simplest track techniques ever introduced in technology. A line.

O.Y.T. is an acronym that stands for On Your Track, what perfectly describes how this device works. Equipped with a set of sensors and some other basic components, this robot is able to follow a line, making corrections to its path according to the direction the line goes. The sensors are strategically positioned to recognize different line patterns and colors, making the possibilities even wider. With such a simple mechanism, this device fulfills its purpose of performing as many different tasks as you can think of, with little investment, since it needs no special tracks or rails, just a simple line on the floor.

This paper will go through the development of this proposed robot, and an explanation of how it works.

# HARDWARE AND CIRCUIT

## The O.Y.T. is composed of two main parts: the robot and the computer station. The robot is elaborated over a PCB and has a small breadboard so it can be attached the ultrasonic sensor, an XBee module and any additional component. There is also a computer with an Arduino MEGA and XBee connected and set up with the MATLAB software. Both parts communicate using XBee protocol.

## Controller

An Arduino Nano was used to control the system because it is compact, and has the same processing power of Arduino Uno or Arduino MEGA. It has all the features needed for this project: enough analog and digital inputs and outputs, PWM and serial communication.

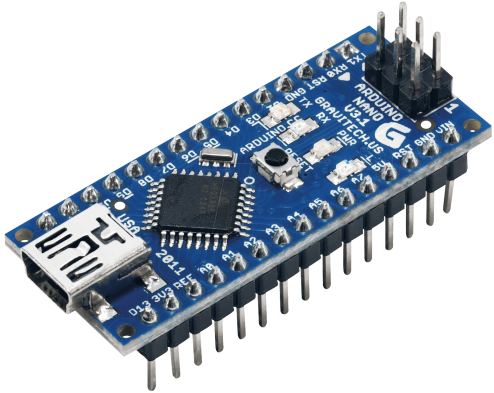


Figure 1 – Arduino Nano

## Infrared sensors

The infrared sensor model used was the CNY70, which is a reflective sensor that includes an infrared emitter and phototransistor in a leaded package which blocks visible light.



Figure 2 – CNY70

Following is a schematic diagram of how the sensor works.

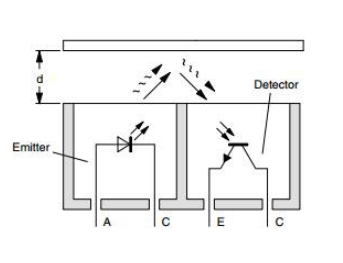


Figure 3 – CNY70 Diagram

Different surfaces reflect different amounts of light. Following is a schematic of how the CNY70 was connected to the circuit.

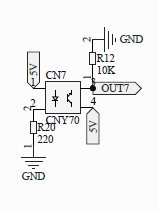


Figure 4 – Circuit of operation CNY70

OUT7 is one of the analog inputs of the Arduino Nano. As the sensor was connected to an analog input of the controller we can use this sensor to identify a lot of different patterns and colors of the surfaces. Depending on the amount of light the surface reflects the impedance of the phototransistor varies. The Arduino measures the voltage drop in the resistor R12.

## Ultrasonic Sensor

An ultrasonic sensor was used to identify obstacles on the path. It was used to avoid collisions, and for security reasons.

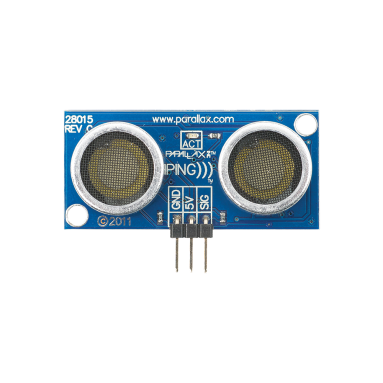


Figure 5 – Ultrasonic Sensor [2]

## DC Motors

To make the robot move was used two DC motors produced by Pololu.

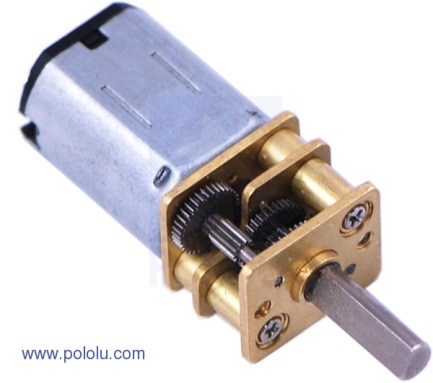


Figure 6 – DC Motor

The motor was connected in a simple chopper circuit that is capable to control the speed of the motors using a PWM signal generated by the Arduino. Following is the circuit schematic.

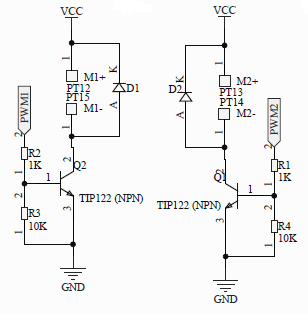


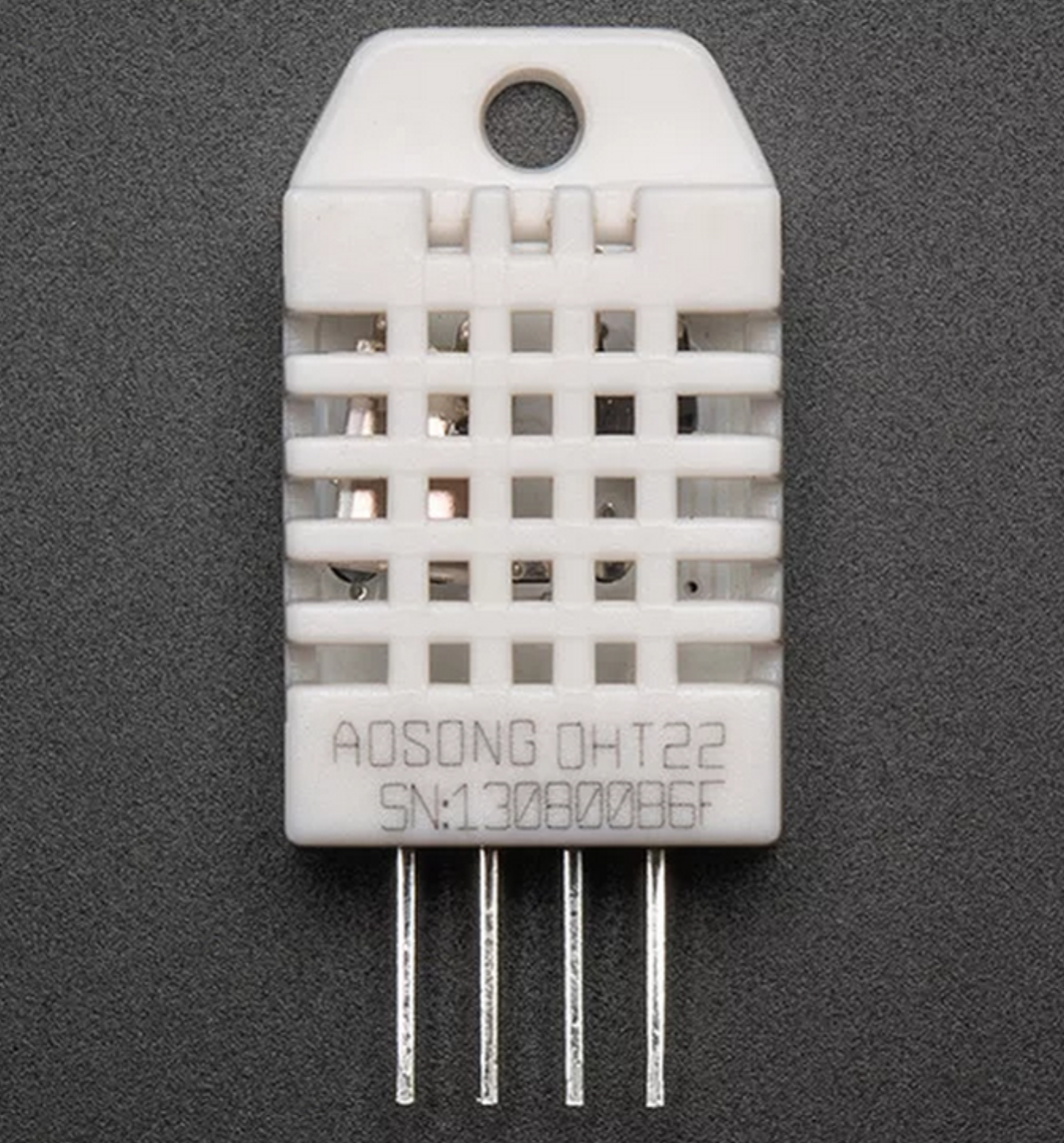
Figure 7 – Circuit of operation DC Motors

The motors are connected in the points M1+, M1-, M2+, and M2-.

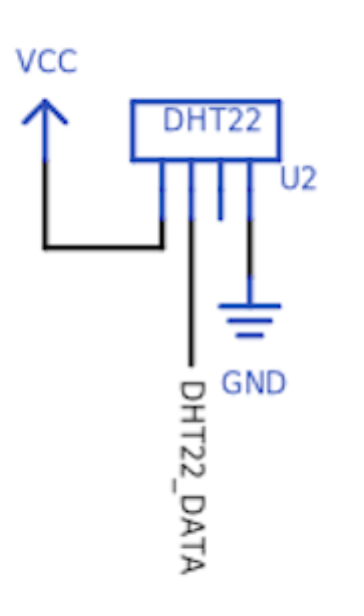
in the points M1+, M1-, M2+, and M2-.

* 1. **Humidity and temperature sensors**

The DHT22 digital humidity and temperature sensor was used for environmental monitoring, using a capacitive humidity sensor and a thermistor. This data is sent to O.Y.T.’s command center (to be covered) through a wireless connection.

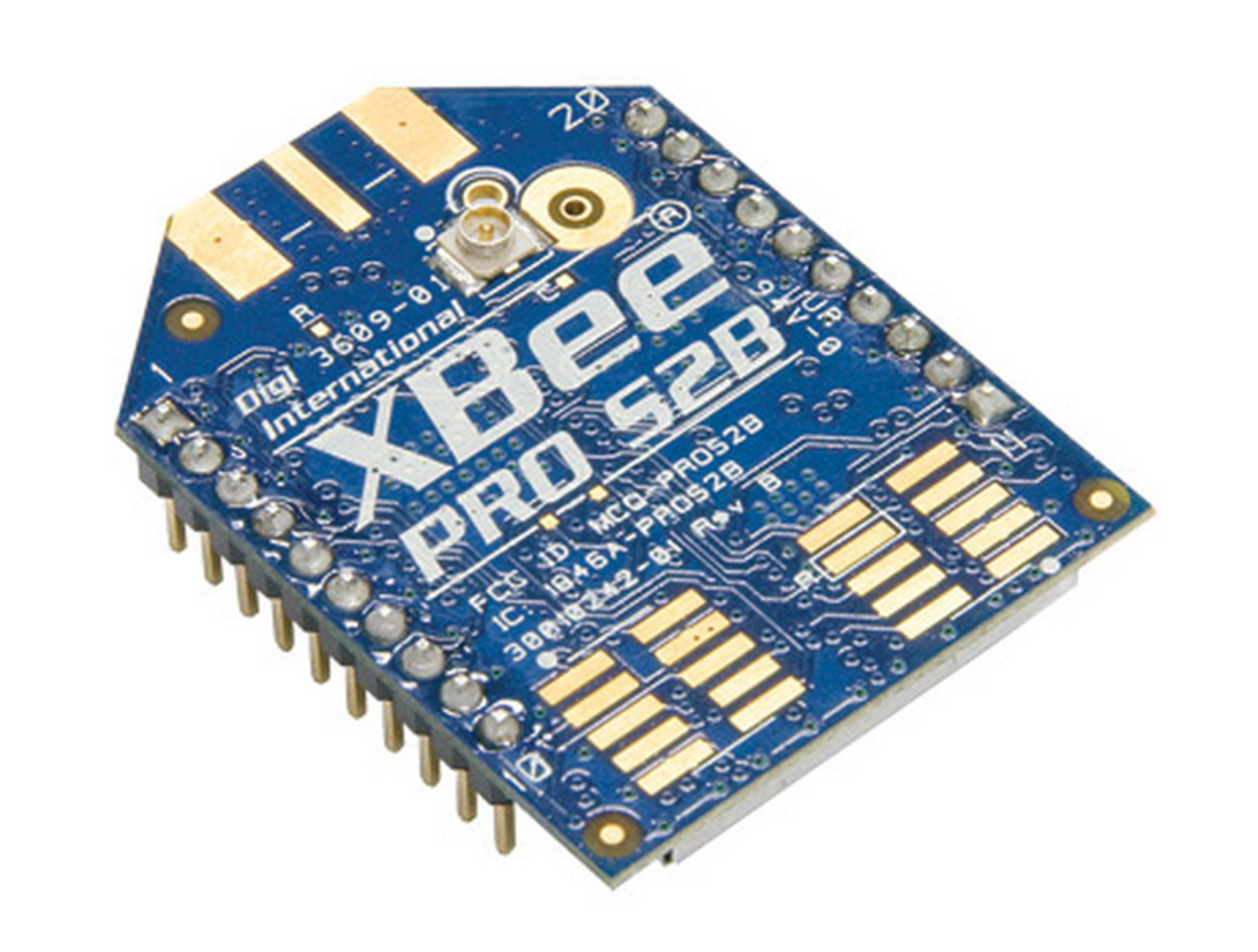
  
Figure 8 – DHT22 sensor. [4]

This sensor’s operation is given by connecting the *data* pin with an Arduino’s data input pin. +5V or +3V are both suitable for power. It also requires a library that can be found in one of the product’s sellers website [3].

  
Figure 9 – DHT22 schematics. [5]

* 1. Communication using XBee

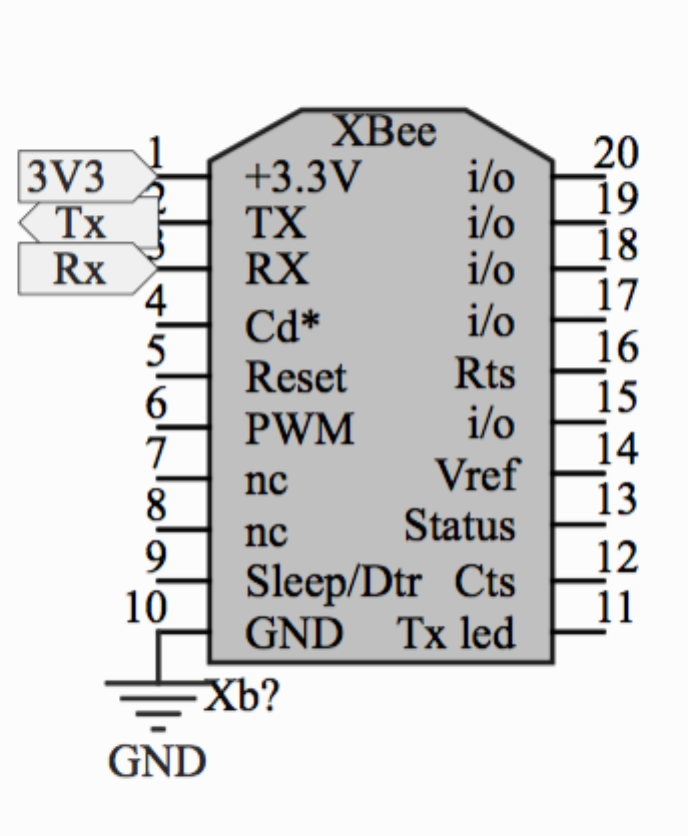
Two *XBee–PRO S2B* were used to establish a PAN network and enable communication between the two Arduinos involved in the process.

  
­Figure 10 - XBee PRO S2B.

Both devices (XBee) must be pre-configured, using appropriate software. In this case, the XCTU was the chosen software. More detailed information about this configuration can be found at the manufacturer website. [6]

The communication process is given by using of the XBee’s TX and RX pins, both connected to the Arduino’s RX and TX pins respectively. +5V or +3V are both suitable for power.

When properly set up, the serial data is transmitted from one Arduino to another through the XBees, at a 2.4GHz frequency, and can be monitored/controlled using the Arduino’s serial monitor. [1]

  
­Figure 11 - XBee PRO SZB schematics.

## PCB Design

A PCB was designed for the main components of O.Y.T. The reasons were to make the circuit reliable and easy to understand. The main intention was to show each part of the circuit in an understandable way.

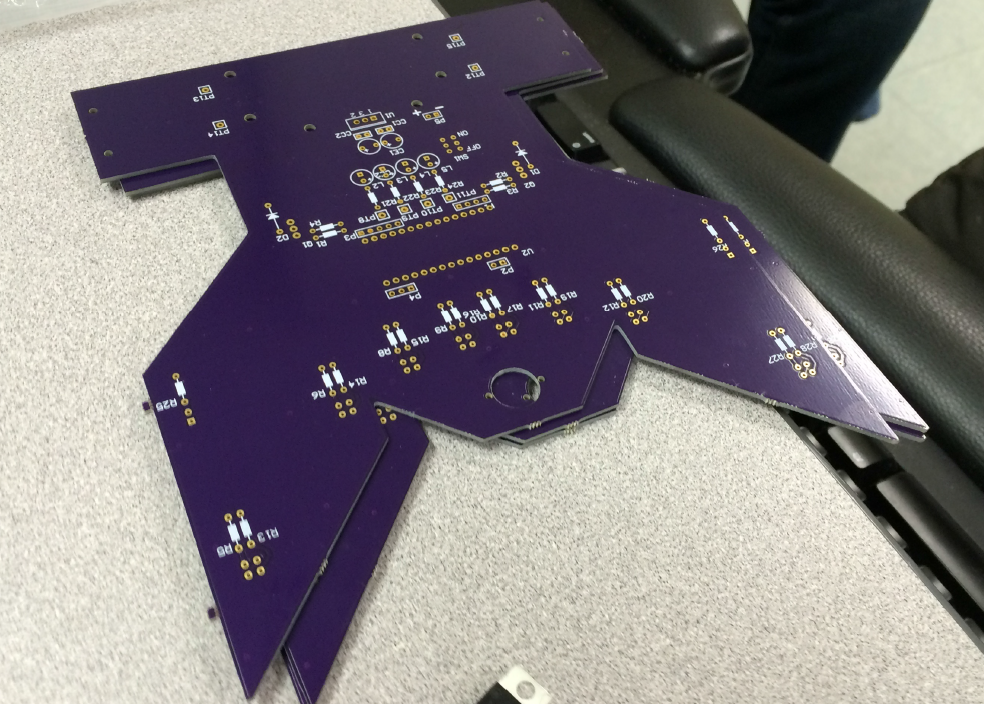


Figure 12 – PCB

# SOFTWARE AND OPERATION

The software of O.Y.T. was built using Arduino platform and MATLAB. It can be divided into two parts: the robot control, the software that controls the Arduino Nano present on the robot, and the command interface, where an Arduino MEGA with XBee makes the bridge between O.Y.T. and a MATLAB program.

# Robot Control

The robot control is the software responsible for making O.Y.T. follow the line correctly. It makes all management of sensors, motors and communication with the command center.

The controller continually reads values from the infrared sensors. In order to reduce noise, it does five readings and take the average in every measurement. As explained in Section 2.2, the Arduino reads a voltage value, based on how much the surface bellow the robot reflects light. In other to determine if the floor is black or white, a fixed threshold was set. If the reading is smaller than the threshold, then it is considered black. Otherwise, white.

Based on the values read from the sensors, the PWM of the DC Motors is changed. This management works like a state machine.

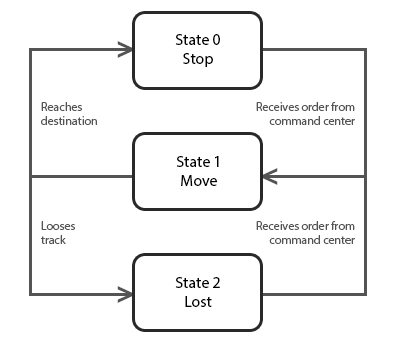


Figure 13 – State Machine

The “state zero” happens if the two sensors on the borders read black. This means that O.Y.T. has found a stop sign. It will stop and send a signal to the command interface indicating it reaches its destination. Along with this signal, it might also send information from the sensors. It can sends temperature information, for example.

The “state one” happens if the robot received an order from the command center and none of the conditions of the other states happens. This is the stage where O.Y.T. follow the track while looking for the destination. If the two central sensors are reading black, the robot should move in a straight line. If only one of the sensors is reading black, it should correct the motors until both are. If any of the other four sensors is reading black, the correction on the motors will be bigger as the distance between the sensor and the center is bigger.

During this state the ultrasonic sensor will be read, and if something is found ahead of the robot, he will stop until the path is clear.

The “state two” happens if all sensors are reading white. This means that the robot has lost the track. The robot will stop and send a signal to the command interface indicating it is lost.

# Command Center

The command center is responsible for sending, receiving, and processing data provided by O.Y.T. system using MATLAB platform. MATLAB has toolboxes that easily set up the communication with Arduino. Arduino MEGA 2560, XBee, and MATLAB 2013a were used to implement the command center. The following picture shows the connection between these three components.

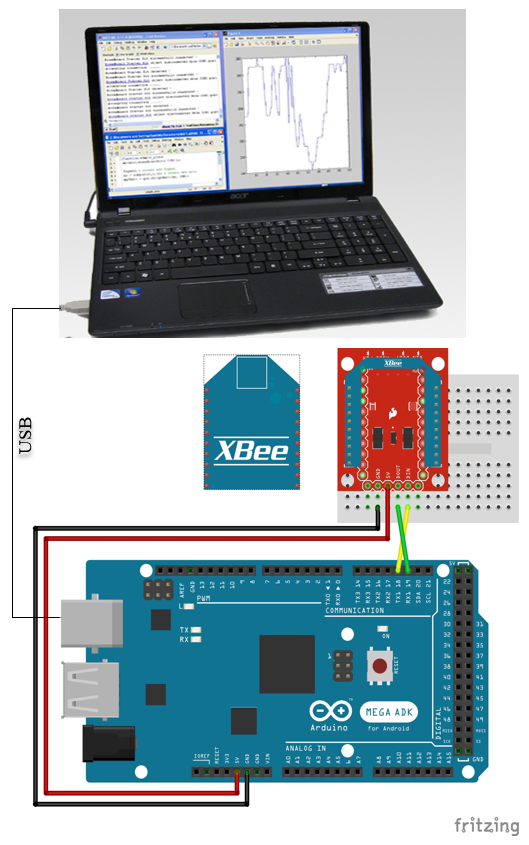


Figure 14 – MATLAB, Arduino MEGA, and XBee

The Figure 15 shows an example of the command center menu.



Figure 15 – MATLAB Program

The “Connect to O.Y.T System” option sets up the serial communication between Arduino MEGA and MATLAB. Using the *serial* command in MATLAB, the programmer is able to set up the Arduino port and baud rate value. If the serial port specified by the programmer is available, the variable portFP holds all details of the new serial port connection. Then, the *fopen* command is responsible for opening the connection between Arduino and MATLAB.

portFP = serial(´COM4´, ´BaudRate´, 9600);

fopen(portFP);

The “START O.Y.T.” and “STOP O.Y.T.” use the command *fwrite* to send data to the Arduino MEGA. When a connection is opened, MATLAB uses the serial ports RX0 and TX0 (USB port) to communicate with the Arduino MEGA, which is responsible for transferring the data to XBee, available in RX1 and TX1 ports.

fwrite(portFP, ‘1’) //Start O.Y.T

fwrite(portFP, ‘0’)//Stop O.Y.T

The “Read Data O.Y.T” option reads data provided by O.Y.T system. Using the command *fscanf,* MATLAB is able to receive 1 byte (8 bits) every time data is available on the serial port. Depending of the data, MATLAB gives a different feedback to the user, such as the messages described on Figure 13. Using the same command, MATLAB can also read data provided by DHT22 sensor and plot graphs.

data = fscanf(portFP);

The “EXIT Application” option closes the command center menu. This option is responsible for closing serial communication using the command *fclose.* In order to release the port to other applications, the command *delete* deletes the serial connection created by the command *serial* in the “Connect to O.Y.T System” option.

fclose(portFP);

delete(instrfind);

# PROTOTYPE BUILT

A prototype was built based on the O.Y.T system description. It is composed by the embedded device and an external MATLAB application. The embedded device is the O.Y.T robot, which is responsible for reading the sensors, figure out the correct way to follow, control the motor speed using PWM, and communicate with the external MATLAB application. The MATLAB application offers a friendly graphical interface that allows the user to send commands to the O.Y.T device and plot real time data.

1. CONCLUSION

The O.Y.T. system shows the potential of an Embedded System. This system proved to be a reliable and an inexpensive way to implement a tracked robot. The Artificial Intelligence implemented in this device is able to adapt to different environment and interpret data provided by an external MATLAB application. This system also shows the potential of a Distributed System because the XBee technology allows data transfer between the O.Y.T robot and an external application running in a powerful computer. Using the Embedded and Distributed O.Y.T System, the user has access to real time data and control.

# ACKNOWLEDGMENTS

Our thanks to Professor Jeremy Hajek and Doctor Daniel Tomal for their help and support in this project.

# REFERENCES

Digi International Inc. (n.d.). Digi. Retrieved 12/2014 from XBee® ZB:

http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/XBee-zb-module#specs [1]

Parallax Inc. (n.d.). Parallax. Retrieved 12/2014 from:

http://www.parallax.com/sites/default/files/downloads/28015-PING-Detect-Distance.pdf [2]

https://learn.adafruit.com/dht/downloads [3]

http://www.adafruit.com/product/385 [4]

https://upverter.com/spark/b18ee98593506a3b/Indoor-Air-Quality-Shield/ [5]

http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/xctu#overview [6]