

SUNY Oswego

**Electrical and Computer Engineering Department**

ECE 492

Follow The Leader Robots

By

Abedallah Abedrabbah and CL

Supervisor

Dr. Mansuer

Spring 2018

**Project Abstract:**

As technology has been vastly increasing over the past decade, robotics and the use of autonomous machines have been starting to be all over the headlines. This idea is about carrying out a complex series of actions automatically that can resemble those of a human being. Engineers are fascinated with the concept of autonomous machines because they can diminish human error which is very efficient and effective. One example that is growing daily and might soon be involved in a wide majority of society, is the idea of creating an autonomous vehicle that will drive and accomplish all the tasks needed to ensure the safety of passengers. This project is based on the basic concepts of autonomous driving and describes how using different motors, sensors, and coding can help accomplish the goal of the project.

This project in total involves three autonomous vehicles, two of which are given commands through wireless communications from the master vehicle on when to perform a certain action to ensure the safety of all three vehicles. The two robots that are not the master will follow the master robot creating a chain of vehicles. The master robot therefore is controlling where the follower robots go. A number of sensors attached to the robotic cars are used for determining what currently is around the robots. These sensors provide information to the robot to perform a task to do in order to adapt to the surroundings safely. On top of each robotic car lies an Arduino Uno microcontroller where the sensors and motors are connected in order to enable the user control over the robotic car to do what task is desired using a variation of coding. The wireless communications is being controlled by Xbee ZigBee modules that are connected on top of each car. The programming language that is used during this project is the software of Arduino IDE which uses C++ commands. These specific functions for the Arduino Uno microcontroller along with the motors and sensors, are used to achieve a follow-the-leader robotic car chain. There are endless uses for a concept like this project in the real world that would lead to more efficient and effective transportation advancements.

**Acknowledgement**

We would like to thank the entire Electrical and Computer Engineering department at SUNY Oswego for their help and support and for their aid they provided to us. Their guidance and constant supervision truly helped the completion of the project. We would like to express our gratitude to Professor Manseur who was our advisor throughout this entire project. We would also like to thank Dennis Quill for all of his advice and suggestions in ordering different components. Lastly, we would like to thank all the other ECE students who helped us with any support that they could. Without all of these people, this project would not have been possible.

**Student Statement**

We as the creators of the Robotic Follow-the-Leader chain ensure that we have applied ethics to the design process and in the selection of the final proposed design. We have held the safety of the public to be paramount and have addressed this in our design.

**Table of Contents:**

* Problem Statement & Formulation Page: 6
* Introduction Page: 7
* Project Specifications Page: 8 - 11
  + Bill of Materials Page: 9
* Detailed Engineering Analysis and Design Presentation Page: 12 - 23
  + Software Page: 13 - 14
  + Motors Page: 14 - 15
  + Line Sensors Page: 15 - 16
  + Ultrasonic Sensors Page: 16 - 18
  + Xbee Modules Page: 18 - 19
  + Shield Schematic Page: 19 - 22
  + Cost Analysis of Bill of Materials Page: 22 - 23
  + Hazards and Failure Analysis Page: 23
* Learning Outcomes Page: 24 - 29
* Conclusion Page: 30
* References Page: 31
* Appendices Page: 32 - 43
* Robot Software Code Page: 34 - 43

**List of Figures and Tables**

***Figure 1:*** *Follow-the-Leader block diagram chain* Page: 8

***Figure 2:*** *Dimensions and layout of robotic vehicle* Page: 10

***Figure 3:*** *Actual picture of the three robot vehicles* Page: 10

***Figure 4:*** *Dimensions and layout of the track* Page: 11

***Figure 5:*** *Pin layout of Arduino Uno* Page: 12

***Figure 6:*** *Design of a two-wire REX Robotics motor* Page: 15

***Figure 7:*** *Sketch of a line sensor reading a light surface* Page: 16

***Figure 8:*** *Illustration of the range the line sensor detects* Page: 16

***Figure 9:*** *Basic layout of how an ultrasonic sensor operates* Page: 17

***Figure 10:*** *Data and Table of the ranges of an ultrasonic sensor* Page: 17

***Figure 11:*** *Image of an Xbee and a Xbee Shield used to mount the Xbee* Page: 18

***Figure 12:*** *Xbee shield schematic with pin connections* Page: 20

***Figure 13:*** *Component connections to Arduino using a shield* Page: 21

***Figure 14:*** *Drawing of component connections to the shield* Page: 22

***Table 1:*** *Cost analysis table of purchased components* Page: 22 - 23

**Problem Statement**

Creating a new concept using autonomous vehicles while ensuring the safety of everyone in the surroundings is a hard task to accomplish. This project is solving the challenge of creating three autonomous robots that will safely complete a course. One robot will be in charge of driving through the course and sending its commands to the two follower robots so they can also finish the course. This will be done by using various sensors and components to help the robots through the course without being harmed.

**Problem Formulation**

There is a constant effort of working with autonomous vehicles in the world. Trying to incorporate new concepts to this idea while keeping the safety of the entire surroundings is the challenge many engineers are facing. With millions of cars on the road and accidents happening every day, this project is designed to address the safety of individuals. A project such as this allows for individuals to get safely where they want on the road. Industries in the shipping workfield could use this concept everyday when needing to ship more than one truck load to a specific place.

1. **Introduction**

Throughout the last decade, the technology of an autonomous vehicle has been advancing greatly. The greatest challenges during this booming concept is ensuring the safety of every individual and making certain that the vehicles are consistent. Nearly 1.3 million people die in road crashes every year globally and additionally over 20 million people are injured or disabled. Road traffic injuries are projected to become the fifth leading cause of death by 2030. Additionally, about 500,000 accidents involve 18-wheeler trucks yearly. Clearly there is an issue here that needs to be solved. Every year further advancements in the autonomous vehicle field are being made in pursuit of lowering this devastating number of injuries.

This project takes an approach to the concept of autonomous vehicles due to wanting to see the number of accidents to decrease in the future. This design in theory will help ensure the safety of individuals on the road. Industries in the shipping workfield could use this concept everyday when needing to ship more than one 18-wheeler truck load to a specific place. In this project, the use of various sensors and components along with knowledge gained throughout engineering courses helped to design three fully functioning autonomous vehicles. The leader robots takes on a course and wirelessly sends the path it is taking to the two follower robots. This will allow all the robots to successfully complete the course in a “Follow-the-Leader” chain. The overall goals to design a project that could help change the world and make the roads a make safer place.

1. **Project Specifications**

The final project was designed to make a group work together to solve a certain problem and to demonstrate the material learned throughout their learning experience as an Electrical and Computer Engineering student. This project will consist of a total of three autonomous robot vehicles. There will be a leader robot whose task is to take on a course and successfully drive through it safely. The leader will also wirelessly send whether it is turning or going straight to the two follower robots and those two robots will execute the tasks that the leader sends to them. All three of these vehicles will have the abilities to stay within a certain course and safely finish without running into each other or any obstacles they might encounter. The robots will begin in a starting box area and navigate its way to the end of the course. It is the job of the master robot to lead itself and the other two robots through any path safely to the finish line. Below is a block diagram of the chain the robots will form during execution of their commands.

***Figure 1:*** *Follow-the-Leader block diagram chain*

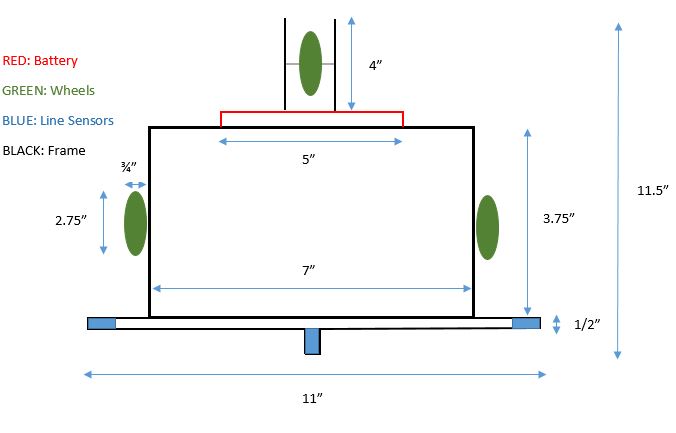
These objectives will be able to be completed with the use of different sensors, components, and programming techniques. Using different sensors to detect objects in the way of the vehicles and sensors to detect if the vehicle is heading off the course such as ultrasonic sensors and line sensors are incorporated to ensure the robots are safe. There will be three line sensors on each vehicle, with them being on the left, right and center. These line sensors were used to navigate the lines to keep the vehicles from not crossing over them to stay on track. The ultrasonic sensor was used to measure distances of an obstacle to prevent the robot from hitting an obstacle. There will be motors attached to the wheels in order to allow the motion of each vehicle. The motors drove shafts and gears to drive all three wheels on the robot. Alongside these sensors and motors, there will be Xbee modules on each robot working together to communicate to each other where the robots should go. The leader robot would use the Xbee module to write and send its data to the two follower robots who would then use their Xbee modules to control where the robot is heading. All of these components are connected together on a designed shield that was connected to an Arduino uno microcontroller. Doing all of this allowed us the ability to control the robot’s movement to complete the task. To have all these components function together, the programming language used was Arduino IDE, which uses commands from C++. Below is a list of all the components incorporated into the design and construction of all the robotic vehicles.

.

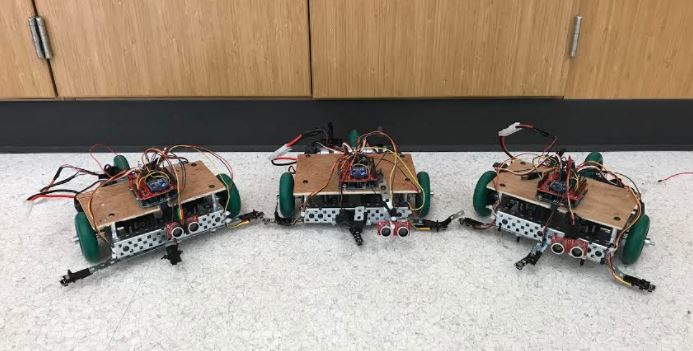
Bill of Materials:

* Arduino Uno Rev3 - Microcontroller [quantity: 3]
* Ultrasonic Sensor - HC-SR04 [quantity: 3]
* 7.2V 3000mAh High Power Battery Pack [quantity: 3]
* Zigbee Xbee module [quantity: 3]
* VEX 2-Wire Motor [quantity: 6]
* VEX Motor Controller [quantity: 6]
* VEX Line Sensor (3-pack) [quantity: 3]
* 2.75” Wheel 4-Pack [quantity: 3]
* Drive Shaft 2” & 3” Pack [quantity: 2]
* Shaft Collar 16-Pack [quantity: 1]
* Nut 8-32 Keps (100-pack) [quantity: 1]
* VEX Tool Kit [quantity: 1]
* VEX Bar 1x25 (8-pack) [quantity: 1]
* VEX C-Channel 1x2x1x35 (2-pack) [quantity: 1]

Each of the robots are designed in a similar fashion taking on the same shape and same components. Having all the robots be built in the same way will help ensure consistency between them when driving the course. The robots all have the same layout of the components and sensors on them. The design for the robot was to keep them small and as compact as possible for better mobility. Below is a picture of the dimensions and layout of each robot car. Also, below is figure 3 which is an actual picture of all three completed robotic vehicles.

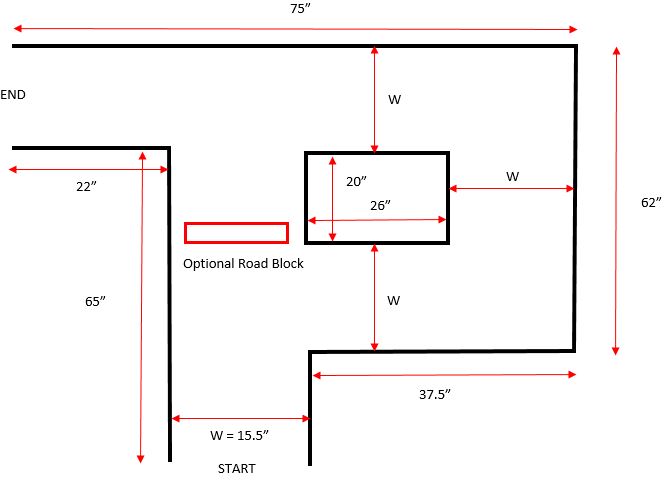


***Figure 2:*** *Dimensions and layout of robotic vehicle*

**

***Figure 3:*** *Actual picture of the three robot vehicles*

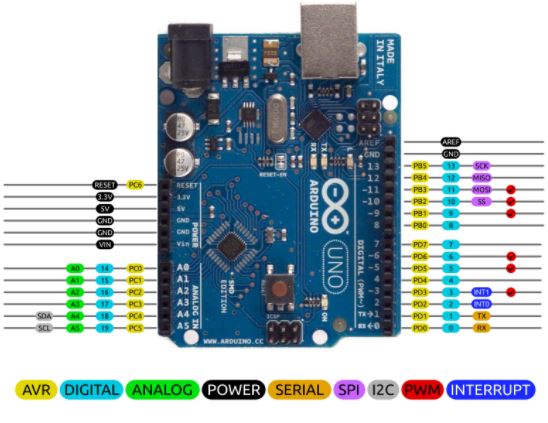
The course is designed to test the robots functionality and the use of all the sensors incorporated on the robots. The robots will have to overcome facing obstacles in the way and successfully turn whichever way they may need. This track will show that the three robots are fully functional of all the demands we asked of them. Below is a diagram of the track that the robots have been asked to successfully complete. The diagram below also shows the dimensions of the track with the bottom opening being the start of the track and the top left opening being the end of the track. An obstacle can be placed blocking the short path and cause the robot cars to take the long loop path to finish the course.



***Figure 4:*** *Dimensions and layout of the track*

1. **Detailed Engineering Analysis and Design Presentation**

For the design process, research had to be done to determine which type of microprocessor would work best for this type of project and which was also relatively cheap. After researching all the different types of microprocessors, the Arduino Uno microcontroller was determined to be the best available for the processes asked of the controller. The Arduino Uno is versatile and can be used to control many different components. In order to maintain the desire to keep the robot vehicles small and compact, the Arduino Uno was perfect for the project. To add on to the benefits this board offered, it was also very cheap compared to some other options. The Arduino Uno was capable of doing everything needed to in order to accomplish the task. Below is a diagram of the pin layout for an Arduino Uno board and will go into detail about different functions on the board itself.



***Figure 5:*** *Pin layout of Arduino Uno*

There are many different pins located on the Arduino Uno with them being power, analog, digital and PWM pins. With the design of this board, a user can directly insert wire right into the black plastic headers to connect to those pins.

* Power: There are different power pins on the board such as 3.3V, 5V and Ground. These pins in Figure 4 are labeled in black.
  + GND: There is three different locations to connect to ground on the board labeled as GND. Any three of these can be used to directly ground the circuit.
  + 3.3V & 5V: The pins labeled as 3.3V and 5V are located on the top left of the board. These two pins will power any of the components the board is running. The 3.3V pin will supply 3.3 volts of power and similarly the 5V pin will supply 5 volts of power when connected to it.
* Analog: All the analog pins on the board are labeled in green and are located on the lower left. These pins will read signals from an analog sensor and convert then to a digital value.
* Digital: All of the digital pins on the board are labeled in light blue. These pins can be used for either a digital input or output.
* PWM: The PWM pins are labeled in red and are located on the right side of the board. These pins are used if you are wanted to use Pulse Width Modulation.

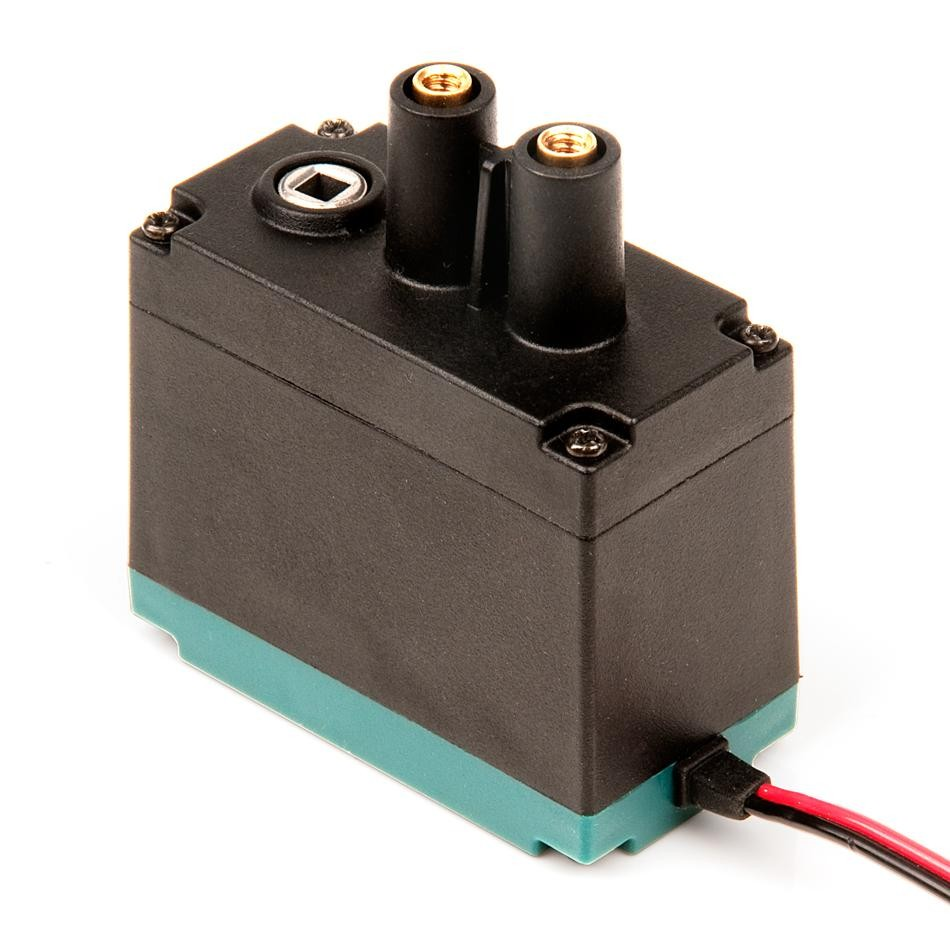
1. **Software:**

For the coding, the software used was the Arduino IDE software which allows the programs to be written in functions of the language C++. This portion of the project took the most time and effort. In order to make sure the robot would complete every part of the task as intended, the process taken was with a step-by-step process. Basically this means writing each part of the code for the individual components and testing to make sure the components were working efficiently and make adjustments accordingly. A goal of the code was to be as consistent as possible. The use of calling methods was a big part in making the program shorter. These methods are used to write a part of the program that the user intends to use more than once and then when you want to perform that specific part, you simple just call it into the main loop function that the robot performs. Some of the main methods needed in the program were moveForward, moveBackward, turnLeft, turnRight, stopMotor, and a few others. Each of these methods had a different task it would perform when it was called upon to the main function that would help the robot complete the task consistently. Two different codes were developed for this project. One of the codes was for the Master robot to drive the course and the other code was for both of the following robots. Incorporate methods for all the sensors to work such as the line sensors and the ultrasonic sensors was needed for these programs to work properly.

The biggest component of the programming was allowing the Xbee modules to communicate and send the code to one another. Writing code that begin the Xbee’s at a specific serial is how they were able to start communicate to one another. Then the Master robot would write to the Xbee’s a different number depending on the task it was performing. The following robots would be asked to read from the Xbee and depending on what number they read, they would perform a task. For example, if the Master robot was to turn right, it would send a ‘2’ to the Xbee. Then the two following robots would read a ‘2’ from the Xbee allowing then know to perform the task asked of them when a ‘2’ is read which is to turn right.

1. **Motors:**

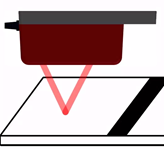
The two VEX motors were attached to the inside of each car with one being on each side. Each motor separately controlled the speed of the car. The motors were a main solution to the project as they were the only devices that actually made the robot mobile. The programming software used was Arduino IDE. This software was able to write the code for the motors to move at the desired speeds. To declare each servo motor in order to move each motor, they were declared as rightServo and leftServo for simplicity. To cause them to move write in the code for example, rightServo.write(80). This would cause the right motor to move clockwise and if you wrote the same thing for the left motor, the car would move forward. In order to have a wheel move backwards, the number inside of that code line would have to be greater than 100. As you assign that number closer to 0, the car moves forward much faster so in order to slow down the motors you would assign the values close to 100. For the motors to move faster in the reverse direction, you would increase the number higher than 100. In order to stop the motors you would simply write a 0 inside both of the servos. By playing around and experimenting with all of these values, it became very easy to move the robot cars around and to allow the robots to turn or move at a certain speed desired. Below is what each of the two-wire VEX robotics motors look like that are located under the base of each of our robots.

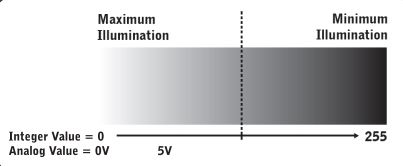


***Figure 6:*** *Design of a two-wire REX Robotics motor*

1. **Line Sensors:**

Without the help of the line sensors, it is very difficult to make the robot complete the task without crossing over a line. To hard code the robot to complete the task would cause very inconsistent values for the robot to perform for the distances and also the robots would have no way to correct themselves. Therefore, it is not very likely the robot would perform up to standards if hard coding was a main piece of the code and also the vehicle would not be very autonomous. To prevent any problems of the vehicles drifting of course, the use of line sensors on our steering wheels and also a central located sensor on the robot. The line sensors have light sensors that give values according to the material color they are reading, which is primarily the difference between light and dark. These sensors are able to tell the difference between a white floor and black electrical tape used for the track. The microcontroller reads these values from a range of about 0 to 255 having the lower the number be a dark surface and the higher the number be a light surface. These values can be obtained experimentally by using the analog to digital converters on the Arduino uno. Once the correct values were obtained and how the line sensors worked, these sensors were incorporated onto the robots which allowed the cars to always stay inside the lines of the track. To call the line sensors in the main function loop, a method had to be written in the program called lineSensorMiddle( ), lineSensorRight( ), and lineSensorLeft( ). Depending on the values obtained by the sensors, the motors would take different actions. There are multiple ways to use these sensors for an autonomous vehicle, but a total of three sensors were used. One line sensor on each side and one in the middle and this worked very effectively. When the left line sensor detects the black line, the motors are set to turn the car right. When the right line sensor detects the black line, the motors are set to turn the car left. When all three sensors did not detect a line, then the robot would drive straight until one of the sensors detected a line. When only the middle line sensor is detected, the motors were set to turn left based on the course our robots were facing. This logic needs to update as the robot moves along the track. The images below show how a line sensor works and the differences that they read when encountering a different surface.

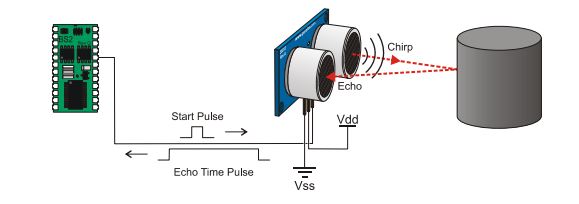


***Figure 7:*** *Sketch of a line sensor reading a light surface*

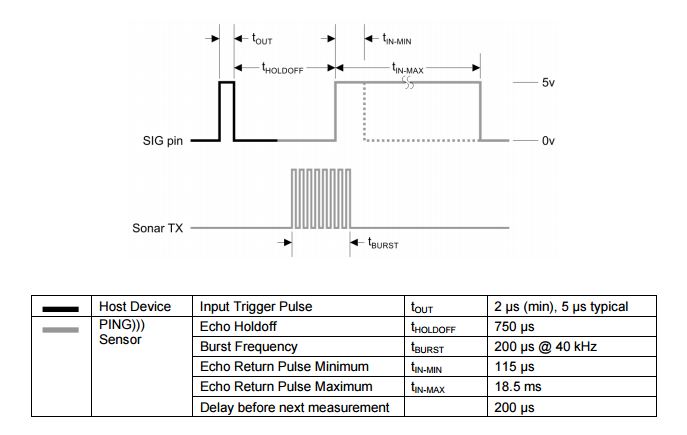
***Figure 8:*** *Illustration of the range the line sensor detects*

1. **Ultrasonic Sensor:**

Ultrasonic sensors were greatly useful in completing the task of having the robot finish the course without hitting any obstacles. These sensors were used to determine if an object was in the path of any of the robots. A ultrasonic sensor provides precise, non-contact distance measurements from roughly two centimeters to three meters. These are very easy to connect to microcontrollers by having to only connect two input/output pin. The sensor works by emitting an ultrasonic sound, well above the human hearing range, and providing an output pulse that corresponds to the time required for the burst of sound to echo back to the sensor. By measuring the echo and trig pulse width, the distance can easily be calculated. These images show ranges for the sensor and how they operate.



***Figure 9:*** *Basic layout of how an ultrasonic sensor operates*

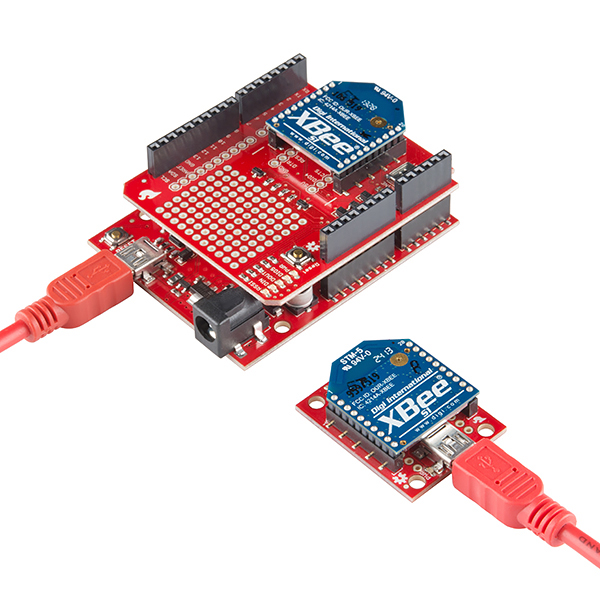


***Figure 10:*** *Data and Table of the ranges of an ultrasonic sensor*

In this project design, the ultrasonic sensor was connected on the front of each robot. This had the intention of helping the robot know when an obstacle was approaching in front of them. In the code, the ultrasonic sensor works off of two methods, the pulse and distance method. The pulse method initiates and determines everything needed for a pulse to be sent out and read. The distance method uses coding lines to determining how far an object is away from the sensor and what actions to have the robot do once this object is detected at a certain distance. Using both of these methods, the ultrasonic sensor was fully functional on the robot, avoiding any obstacle that it encountered. If an object was detected by the sensor at a certain distance, than the robot turned right to avoid the object. Having the robot turn right successfully had the vehicle avoid the object and also maintain inside of the track. The ultrasonic sensor will work how ever many times an object is encountered during the course of a track.

1. **Xbee Module:**

The Xbee was used in the project to communicate between the Arduino Uno boards wirelessly. This would allow the master robot to communicate to the boards of the two follower robots what task to perform. Below is a picture of the Xbee module along with the Xbee shield.

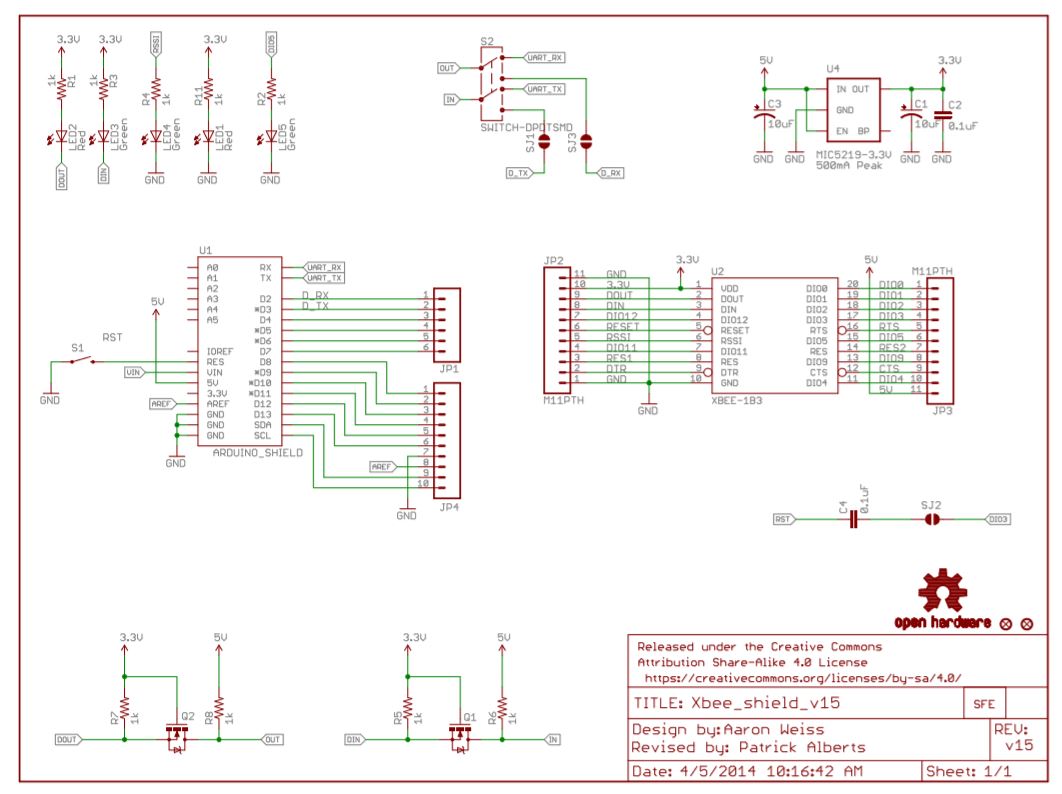


***Figure 11:*** *Image of an Xbee and a Xbee Shield used to mount the Xbee onto the Arduino Uno*

The Xbee and the Xbee shield were both used in this project in order to connect the Xbee to the Arduino Uno and to connect all the other sensors together. The XCTU program provided by Digi the manufacturer was used to program the Xbees. All Xbees where updated with the DigiMesh Network firmware where we configured the Channel to 0x10 and the network ID as 0xA4A4 for all three xbees. The only difference between the Xbees was the master robot’s was configured as a coordinator mode and the other two as standard routers. Also they were named Coord for the coordinator and R1 and R2 respectively. The coordinator was set to broadcast its signal while the routers were set to send information to the coordinator only. The coordinator sends single number which the routers define each number for a certain task. The number zero for moving forward, one to stop the robot, two to go left and three to go right.

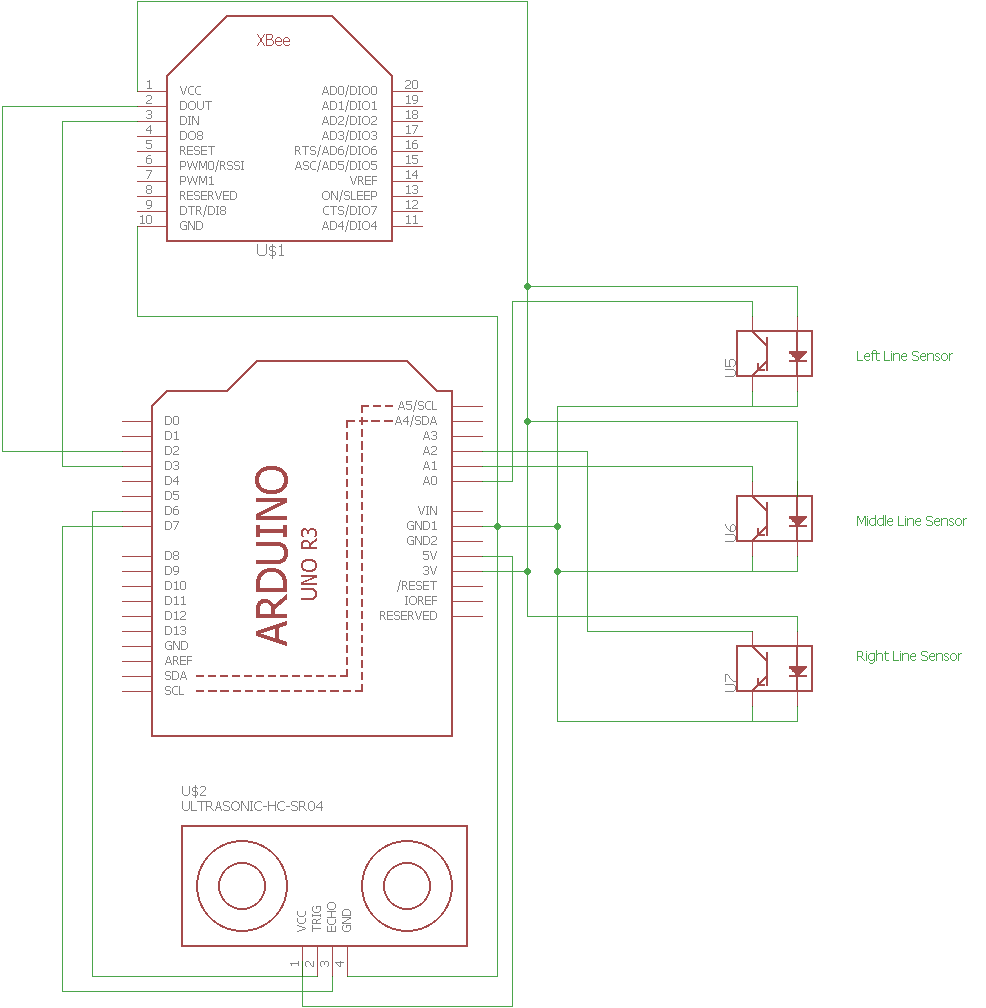
1. **Shield Schematic:**

The Xbee shield was used as the base of connecting the components to the arduino and powering them. Xbee power pin is connected to 3.3V and ground to the ground pin of the arduino. The shield switch is set to dline or digital line position which lets us to use the IO pins 2 for DOUT which is acting as the Tx or transmission pin and 3 for DLINE which is acting as Rx or receiving pin. This is due to the xbee UART mode messing up the Arduino’s communication with the Arduino IDE and also in UART mode the XBee would cause constant interrupts in the arduino making them incapable of moving and functioning correctly. Below is the Xbee shield schematic showing which pins on the Xbee is connected to on the shield

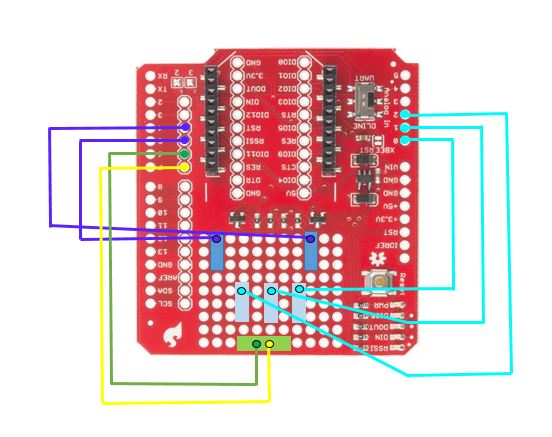


***Figure 12:*** *Xbee shield schematic with pin connections*

The next schematic show how we connect the components to the arduino using the shield when building our robots. The line sensors are connected to the analog pins. Each robot has a left, middle, and right line sensors connected to Analog pins 0, 1, and 2 respectively. The ultrasonic sensor is connected to pin 6 for the pulse modulation output and pin 7 for the echo pin for if an obstacle is detected. Figure 14 provides where each wire was connected onto the shield. The wiring includes all of the sensors used such as the line sensors, ultrasonic sensor, and the two-wire motors.



***Figure 13:*** *Component connections to Arduino using a shield*

**

***Figure 14:*** *Drawing of component connections to the shield*

1. **Cost Analysis of Bill of Materials**

Included below is a cost analysis of everything purchased for this project. This includes a list of the parts for the design that were used and every part is identified by the name, the part number is based on the seller of the product. The cost for each of these items is shown per unit cost and also the quantity purchased.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Part | Part Number | Distributor | Quantity | Price | Total |
| Ultrasonic Sensor HC-SR04 | 1568-1421-ND | DigiKey | 3 | $3.95 | $11.85 |
| Line Sensor (3-pack) | 276-2154 | VEX | 3 | $39.99 | $119.97 |
| 2-Wire Motor | 276-2177 | VEX | 6 | $14.99 | $89.94 |
| Motor Controller | 276-2193 | VEX | 6 | $9.99 | $59.94 |
| Drive Shaft 2" & 3" Pack | 276-2011 | VEX | 2 | $5.49 | $10.98 |
| Shaft Collar (16-pack) | 276-2010 | VEX | 1 | $7.99 | $7.99 |
| Arduino Uno Rev3 | 1050-1124-ND | DigiKey | 3 | $26.13 | $78.39 |
| 2.75" Wheel (4-pack) | 276-1496 | VEX | 3 | $9.99 | $29.97 |
| Nut 8-32 Keps (100-pack) | 275-1026 | VEX | 1 | $2.99 | $2.99 |
| Tool Kit | 276-1382 | VEX | 1 | $2.99 | $2.99 |
| Bar 1x25 (8-pack) | 275-1141 | VEX | 1 | $12.99 | $12.99 |
| C-Channel 1x2x1x35 | 276-2906 | VEX | 1 | $8.99 | $8.99 |
| 7.2V 3000mAh Battery Pack | 11204-01 | Tenergy | 3 | $32.98 | $98.94 |
| Zigbee Xbee Module | 602-1557-ND | DigiKey | 3 | $17.50 | $52.50 |
|  |  |  |  |  |  |
|  |  |  |  | TOTAL | $588.43 |

***Table 1:*** *Cost analysis table of purchased components for entire project*

1. **Hazards and Failure Analysis**

As Electrical and Computer Engineering students at the State University of Oswego, the design problem ensures the ethical responsibility. The solution to the design problem is safe to the public and the environment. This is substantiated by showing that the Design for Safety methods were employed in the design process. The project ensures the safety by individually testing all the components of the design and ensuring they work properly with all of the required specifications. All of the components work properly and up to safety standards and is safe to the public and the environment.

1. **Learning Outcomes**

This project has applied each of the Department’s Learning outcomes in the design and development of the entire project. These outcomes are very important for every engineer to understand and the project incorporated each one to some extent. This project incorporates IEEE standards which are rules that are set worldwide for engineers to follow in all of the designs created. IEEE is a technical professional association that has numerous standards to promote growth and interoperability of existing and emerging technologies. The IEEE standards ensure that products and services perform as intended.

An IEEE standard that was used for the design of this project was the standard of wireless communication. The Xbee’s use this standard for transmitting data over a preset range of frequencies. This standard followed by the Xbee’s allow them to be able to handle all the communications between the devices and will be simpler to implement than trying to use a protocol with mesh to accomplish the same goal. This standard is known as 802.15.4 and is a standard for wireless communication by the IEEE.

Another IEEE standard that was used for various components in this project was the letter symbols for units of measurements. This standard focuses on the use of SI units being clearly identified. The measurements such as distance for the ultrasonic sensor is a key component that had to follow this standard. The ultrasonic sensor converts time of a wave sent out into distance. Using distance, the standard makes it clear that a unit of measurement must be shown. If there was no use of units in any measurement then the calculation would never be clear as to what the actual measurement is.

1. **ability to apply knowledge of mathematics, science and engineering**

Mathematics, science, and engineering is the background to any Electrical and Computer Engineer. This project incorporated knowledge of all three subjects. Mathematics was incorporated in this project with the use of the ultrasonic sensors. In order to receive a precise distance of an object, you have to use a calculation formula to indicate the distance in centimeters. Having the knowledge of engineering was very important in this project. The uses of engineering knowledge in this project was endless. We had to design and build our own robots, set up all the components and sensors, wire together the shield on top of the Arduino, program the code for everything to work together, and much more. All of these applications were the basis for designing and building this project. All the courses that we took throughout our degree taught us a great amount of mathematics, science, and engineering which helped our group complete our project. These courses allowed for us to be successful in coming up with our own project and delivering that project.

1. **ability to design and conduct experiments, as well as to analyze and interpret data**

For this project we needed to be familiar with all the components we were going to implement and we needed to know how to go about creating a successful project. We had to conduct many experiments with each individual component of our project to see how that part functions and how to make it work correctly. In order to do this we had to test out different codes, wiring, and power to see what worked best for each part. We had to analyze and interpret data constantly to see what would work best in our design. Also all of this was very important in figuring out what we exactly needed to use. This knowledge came from many key courses we have taken where the use of hands on labs help students have a very good understanding of how to design and conduct experiments. To go along with this, being able to conduct an experiment is practically useless if you cannot analyze and interpret the data that you get from it. This outcome was used throughout our project and helped us be able to be successful when designing all the key components.

1. **ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability**

We had to come up with a project idea that was accepted by our engineering faculty and then had a certain timeline to perform certain tasks and be able to deliver the project at a specific date. We had to design a system with many different components to meet a desired need. That need had to be cost efficient to that we could afford with our department aid. We had to meet the need of safety which is a big concern because the safety of individuals is the most important thing. With this project, our group was able to successfully show the ability to design a system and components to meet the desired needs no matter what they entailed.

1. **ability to function on multidisciplinary teams**

This project was a group effort and therefore we had to showcase our ability to function in teams. Our group of two that was together for this project never worked together in the past and we worked very well from day one. Constantly communicating with each other, helping each other with anything the other might need. Without this good partnership this project would have been a very difficult task to overcome. We worked together designing and building the robot as a whole. We research and tested all the individual components together so we both had a very good understanding of how we were going to take on this task. We both taught each other things we did not understand and ran ideas by each other to try to improve everything we created. Working together as a team and being able to get all the tasks done before a specific deadline is something our group showed we could do during this project. All together this entire project was a group effort and truly shows our ability to function on multidisciplinary teams no matter the challenges faced.

1. **ability to identify, formulate, and solve engineering problems**

We designed this project and had to identify and formulate all of the pieces of it. We had to solve many engineering problems about how to design or how to program all of the pieces we were using. Some of the components we did not use before and we have to figure out a way to solve these problems through various research and testing. If a component did not seem to be working up to its standards, our group was able to identify this problem and find a better solution to completing the task of finishing the track. Designing the entire robot from scratch and discovering which components we needed to use to help us best be successful at this project is a big way we showed this objective. Certain times throughout testing the robot something would not be working correctly on the vehicle. It was up to us to figure out what piece was malfunction and formulate a solution to the problem. We have to solve many problems throughout the entire project in order for each car to better complete the track.

1. **understanding of professional and ethical responsibility**

An engineer has the responsibility is to keep the public safe and to create a design that could harm individuals is very unprofessional and not ethical. Therefore being responsible engineers, our group made sure to be both professional and ethical. We made sure to test all the components to ensure the safety of every part of our project. We ran multiple tests on the track to show that the project as a whole is harmful to anyone in the surroundings. Also a big part of this learning outcome is policies such as academic dishonesty. We insure our honesty into this project and are very proud of our accomplishments made during this project.

1. **ability to communicate effectively**

An engineer could have the greatest idea in the world, however if that person cannot communicate effectively, nobody is going to support that idea. Our group constantly communicated effectively to one another and also to the Electrical and Computer Engineering faculty at SUNY Oswego. We were able to communicate effectively our project as a whole from what we wanted to accomplish to everything we needed for the project. Communicating effectively to each other in our group was very important to getting the project done. We talked daily as to what we felt needed to be worked on in order to get everything done by the deadline. We also communicated to each other any issues or misunderstandings that we had and worked everything out to successfully complete the project to the best of our abilities. We communicated effectively to our faculty with things such as components we had to order and how we wanted to approach different parts of the design. They gave us their feedback and help and that greatly progressed us along during the course of this project. Without communicating well to each other and our faculty, this project could have faced many more issues and might not have been completed.

1. **the broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and social context**

Engineers have to understand how different designs can affect the environment and global and social context. We accomplished this during the course of our project by constant research of how autonomous vehicles are affected the world. There is constant money being poured into the design of this technology with the idea to make it as safe and efficient as possible. The design of autonomous vehicles will continue to affect the global context as it continues to improve and be seen more and more in everyday lives. Our project has to do with the technology of autonomous vehicles and we see the huge upside to this technology and see the environment being safer with more of these on the roads once the technology has been improved.

1. **recognition of the need for, and an ability to engage in lifelong learning**

With the vast improvements and constant change in technology, engineers would be left back in time if they were persistent on continuous learning. This project expanded our learning dramatically, working with various components that we did not have prior experience with. Our group saw the recognition for further learning and researched for long periods of times every component we were working with. We researched different approaches we could take as well and focusing on which ones we thought would help us best accomplish our task. This project taught us how to work with a lot of different softwares and components that we were not familiar with and how to approach learning how to use them properly. We see the huge importance in engaging in lifelong learning being engineers we see everyday how technology is changing and improving. We want to always be up to date with the newest and best technologies out there. Therefore we incorporated this idea into our project to learn about different components that we thought would work best for us with the task in mind.

1. **knowledge of contemporary issues**

Contemporary issues can influence or change code of ethics and could change how things would have to be done. Knowing of issues happening currently in the world such as problems other engineers had trying to design something can help save time and learn from those issues. This learning outcome I feel we incorporated the least into our project. I believe we incorporated this into our project anytime we researched on autonomous vehicles and reading into what the new issues with accidents or new technologies have been occuring. There have been big headlines in the news about these self-driving cars getting in serious accidents and that immediately scares the public. Due to knowing these contemporary issues, engineers are constantly trying to fix these malfunctions to have their designs the most efficient and effective they can be.

1. **ability to use the techniques, skills, and modern engineering tools necessary for engineering practice**

Throughout the key courses we learned in the curriculum at SUNY Oswego, a variety of techniques, skills, and modern engineering tools to help in engineering practice. These tools all included testing and using different devices and components. Our project has multiple different sensors and a microcontroller that we had to use tools to test every part of them to ensure we were using them correctly. Every test we did on our robotic cars and its pieces were skills that we had acquired through all of our engineering practices. Our ability to use these tools help us overcome any problems we faced during this project and helped us reach our goal. Modern engineering tools will continue to change and improve and being able to stay up with them is very important in all engineering practices.

**Conclusion:**

The overall solution was well executed to the best of our knowledge to make all three robotic cars complete the task of the track. The leader robot was able to navigate through the course without hitting any obstacles and staying within the track while also being able to send the commands it was executing to the following robots. The following robot was able to execute the commands sent from the leader robot in order for the follower to safely make their way to the finish line. The main job to keep the robot in between the lines was done by the line sensors. These line sensors were placed on each side of the car and one in the middle. The line sensors would adjust the car back onto tack when any of them were activated. These line sensors worked very well for this task. Using the motors to drive the car in any direction worked very well and enabled the robots to continue to move to the finish line. Fully charging the batteries constantly ensured that the batteries were allowing each vehicle to perform consistently as possible. The ultrasonic sensor was used to detect obstacles at a certain distance and to navigate the robot cars away from those obstacles. This sensor worked very well in detecting obstacles in the way of the robots and successfully would turn the cars away from them. The Xbee modules were used for communicating between each of the Arduinos on the robotic cars. These modules worked very well in detecting when a certain command was happening to the master robot and letting the follower robots know which task to perform. Overall, using the motors, sensors, and Xbee’s attached to the robot and through a lot of test and error, the project was a success and our robot cars were able to efficiently drive safely through the paths it took to the finish line.

**References:**

For more information on the project design please refer to these links for further learning.

1. “Road Crash Statistics,” *ASIRT*. [Online]. Available: <http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics>.
2. “Bouniarduino Pinout Libraries Io,” *Bouniarduino Pinout Libraries Io Arduino Uno Pinout Voltage Arduino Pinout*. [Online]. Available: <http://tsulo.com/g/0334_012/>.
3. “VEX 2-Wire Motor 393,” *Robot Mesh*. [Online]. Available: <https://www.robotmesh.com/vex-robotics/vex-2wire-motor-393>.
4. M. #385612, “SparkFun XBee Wireless Kit,” *KIT-13197 - SparkFun Electronics*. [Online]. Available: <https://www.sparkfun.com/products/13197>.

1. <https://content.vexrobotics.com/docs/instructions/276-2155-instr-0312.pdf> [Ultrasonic Sensor]

1. <https://www.parallax.com/sites/default/files/downloads/555-27401-QTI-Line-Sensor-Documentation-v1.0.pdf> [Line Sensor]

1. <https://content.vexrobotics.com/docs/inventors-guide/2-wire-269-inst.pdf> [Two-wire motor]

1. [https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-zigbee#specifications](https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-zigbee" \l "specifications) [Xbee specifications]

1. <https://store.arduino.cc/usa/arduino-uno-rev3> [Arduino Uno]

**Appendices:**

This section includes specifications for using the components incorporated into this project correctly and safely. This also includes the guidelines this project followed to ensure the components were safe to the environment. Also this section includes the code written in Arduino IDE for both the master robot and the follower robot.

**Appendix 1: Arduino Uno Rev3 Microcontroller**

A microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. The length is 68.6 mm, width is 53.4 mm, and the weight is 25 g.The technical specifications for operation this microcontroller are as stated:

* Operating Voltage: 5 V
* Input Voltage (Recommended): 7-12 V
* Input Voltage (Limit): 6-20 V
* DC Current per I/O pin: 20 mA
* DC Current for 3.3V Pin: 50 mA
* Flash Memory: 32 kB of which 0.5 kB used by bootloader
* SRAM: 2 kB
* EEPROM: 1 kB
* Clock Speed: 16 MHz

**Appendix 2: Ultrasonic Sensor - HC-SR04**

The ultrasonic ranging sensors provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Includes an ultrasonic transmitter, a receiver, and a control circuit. There are four pins: VCC(Power), Trig(Trigger), Echo(Receiver), and GND(Ground). The dimension of the sensor is 45\*20\*15 mm.

* Operating Voltage: 5 V DC
* Operating Current: 15 mA
* Measure Angle: 15 degrees
* Ranging Distance: 2 cm - 4 m
* Working Frequency: 40 Hz

**Appendix 3: Zigbee Xbee Module**

The Xbee module has a supply voltage of 2.1 to 3.6 V. Transmit current standard of 33 mA at 3.3 VDC/ 45mA boost mode. Transmit current programmable of 47 mA at 3.3 VDC/59mA boost mode. CPU/Clock speed up to 50.33 MHz. The Xbee has a serial data interface UART and SPI. It has an indoor range of up to 200 feet and outdoor range of up to 4000 feet. A data rate of RF 250 Kbps.

**Appendix 4: VEX 2-Wire Motor**

The 2-Wire Motor 393 provides 60% more output power than the standard motor, which allows for more powerful mechanisms and drive bases. The motor connects directly to 2-Wire Motor ports.

* Output Speed (RPM): 100
* Ticks per Revolution: 627.2
* Stall Torque: 1.67 N-m

**Appendix 5: VEX Line Sensor**

Consists of an infrared light sensor and an infrared LED. It functions by illuminating a surface with infrared light; the sensor then picks up the reflected infrared radiation and, based on its intensity, determines the reflectivity of the surface in question. Lightly colored surfaces will reflect more light than dark surfaces; therefore, lightly colored surfaces will appear brighter to the sensor. This allows the sensor to detect a dark line on a pale surface, or a pale line on a dark surface. The range for the Line Tracker is approximately 0.02 to 0.25in (from the ground) with optimum sensitivity at 3 mm (about 1/8 inch). The minimum line width it can detect is 0.25in. Connect to three analog inputs

* Black: ground
* Red: +5V
* White: control signal

**Appendix 6: VEX Motor Controller**

The motor controller regulates the speed of a VEX motor based on a signal it receives from a microcontroller. Max motor stall current is 3A. The max battery voltage is 8.5V. The inputs of the motor controller are:

* Black Wire - Ground
* Orange Wire - Power
* White Wire - PWM Signal: 1ms - 2ms will give full reverse to full forward, 1.5ms is neutral

The output is Red & Black Wires - Motor Connections.

**Appendix 7: Arduino IDE Software Code**

**Appendix 7a: Master Robot Software Code**

// Program for the Master Robotic Vehicle

// This program enables the master robot to drive through a given course safely and

// writes/sends the commands the vehicle is performing to the follower robots.

// These are libraries used for the Arduino microcontroller

#include <Servo.h>

#include <SoftwareSerial.h>

#include <digitalWriteFast.h>

// XBee's DOUT (TX) is connected to pin 2 (Arduino's Software RX)

// XBee's DIN (RX) is connected to pin 3 (Arduino's Software TX)

SoftwareSerial Xbee(2, 3); // TX, RX

#define echoPin 7 // Echo Pin

#define trigPin 6 // Trigger Pin

#define LEDPin 13 // Onboard LED

Servo rightServo; //Initialize the right motor

Servo leftServo; //Initialize the left motot

//To get information from Xbee

char received;

//For Motor

int angle = 0; // Servo position in degrees

int count = 0;

//For Ping Sensor

int maximumRange = 21; // Maximum range needed

int minimumRange = 0; // Minimum range needed

double duration; // Duration used to calculate distance

int distance; // Distance used for ping sensor

//For Line Sensors

int sensorMiddle; // reads the input on analog pin 1

int sensorRight; // reads the input on analog pin 2

int sensorLeft; // reads the input on analog pin 3

void setup()

{

rightServo.attach(5); // pin 5 is connected to the right motor

leftServo.attach(4); // pin 4 is connected to the left motor

Serial.begin(9600);

Xbee.begin(9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(LEDPin, OUTPUT);

pinMode(echoPin, INPUT);

} //End Setup

void loop()

{

distance = ObjectPresent(); // Gets value of ultrasonic sensor

sensorMiddle = analogRead(A1); // reads the input on analog pin 1, middle line sensor

sensorRight = analogRead(A2); // reads the input on analog pin 2, right line sensor

sensorLeft = analogRead(A0); // reads the input on analog pin 3, left line sensor

if (distance > maximumRange && distance > minimumRange) // If no object is in the way of the vehicle

{

if (sensorMiddle < 600 && sensorLeft < 600 && sensorRight < 600) // If all line sensors are touching a white floor

{

moveForward(); // Move forward

}

else if (sensorMiddle > 600) // Line sensor middle hit line

{

lineSensorMiddle(); // Perform the line sensor middle method

}

else if (sensorRight > 600) // Line sensor right hit line

{

lineSensorRight(); // Perform the line sensor right method

}

else if (sensorLeft > 600) // Line sensor left hit line

{

lineSensorLeft(); // Perform the line sensor left method

}

}

if (sensorLeft >= 600 && sensorRight >= 600) // If the left and right line sensor are on a black line

{

stopMotor(); // Stop the robot

}

else if (distance < maximumRange && distance > minimumRange) //Object too close to vehicle

{

rightAngle(); // Robot turns right to avoid object

}

} //End Loop

//------------------------------------------------------------------------------------------

// Methods to call during loop

//------------------------------------------------------------------------------------------

void moveForward() //Drive forward

{

rightServo.write(84); // Set speed of right motor

leftServo.write(84.8); // Set speed of left motor

delay(100);

Xbee.write('0'); // Write a '0' to the Xbee for the following vehicles to read

}

void moveBackward() //Drive backwards

{

rightServo.write(105);

leftServo.write(105);

delay(100);

}

void turnRight() //Turn car right

{

rightServo.write(105);

leftServo.write(70);

delay(50);

}

void lineRight()

{

rightServo.write(75);

leftServo.write(0);

delay(30);

}

void lineLeft()

{

rightServo.write(0);

leftServo.write(75);

delay(30);

}

void turnLeft() //Turn car left

{

rightServo.write(65);

leftServo.write(0);

delay(50);

}

void stopMotor() //Stop the car

{

Xbee.write('1'); // Write a '1' to the Xbee for the following vehicles to read

rightServo.write(0);

leftServo.write(0);

delay(30);

}

void lineSensorMiddle() // Line sensor middle stops the car

{

for (angle = 0; angle < 6; angle++) // Tells robot how long to perform task

{

moveBackward(); // Move vehicle backwards

}

leftAngle(); // Turn left

delay(50);

}

void lineSensorRight() //Line sensor right turns car left

{

for (angle = 0; angle < 0.5; angle++)

{

lineRight(); // Perform line right method

}

}

void lineSensorLeft() //Line sensor left turns car right

{

for (angle = 0; angle < 0.5; angle++)

{

lineLeft(); // Perform line left method

}

}

void rightAngle(){

for (angle = 0; angle < 13; angle++)

{

turnRight(); // Perform turn right method

}

delay(50);

Xbee.write('3'); // Write a '3' to the Xbee for the following vehicles to read

}

void leftAngle(){

for (angle = 0; angle < 11.5; angle++)

{

rightServo.write(70);

leftServo.write(105);

delay(50);

}

delay(50);

Xbee.write('2'); // Write a '2' to the Xbee for the following vehicles to read

}

int ObjectPresent() // For ultrasonic sensor to determine where the object is

{

analogWrite(trigPin,175);

duration = pulseIn(echoPin, HIGH);

return duration / 58.2;

}

**Appendix 7b: Follower Robot Software Code**

// Follower Robot Code

// Program to drive the follower robot through the course successfully by receiving commands from master

#include <Servo.h>

// We'll use SoftwareSerial to communicate with the XBee:

#include <SoftwareSerial.h>

SoftwareSerial XBee(2, 3); // TX, RX

#define echoPin 6 // Echo Pin

#define trigPin 7 // Trigger Pin

#define LEDPin 13 // Onboard LED

char MasterControl;

Servo rightServo; // Initialize right motor

Servo leftServo; // Initialize left motor

//For Motor

int angle = 0; // servo position in degrees

//For Ping Sensor

int maximumRange = 15; // Maximum range needed

int minimumRange = 0; // Minimum range needed

double duration; // Duration used to calculate distance

int distance;

//For Line Sensors

int sensorMiddle; //reads the input on analog pin 1

int sensorRight; //reads the input on analog pin 2

int sensorLeft; //reads the input on analog pin 1

int counter;

int count;

void setup()

{

rightServo.attach(5); //pin 5

leftServo.attach(4); //pin 4

XBee.begin(9600);

Serial.begin(9600);

counter = 0;

count = 0;

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(LEDPin, OUTPUT);

} //End Setup

void loop()

{

distance = ObjectPresent();

sensorRight = analogRead(A2); //reads the input on analog pin 2, line sensor right

sensorLeft = analogRead(A0); //reads the input on analog pin 1, line sensor left

//Robot is on track, drive straight

MasterControl = XBee.read();

if(MasterControl == '0') // Xbee reads a '0' from master robot

{

if (sensorRight < 600 && sensorLeft < 600) // If line senor left and right are touching white floor

{

delay(105); //Delay for first right turn

moveForward(); //Move forward

}

else if (sensorRight >= 600) // Line sensor right hits line

{

lineSensorRight(); // Perform right line sensor method

delay(70);

moveForward(); // Move forward

}

else if (sensorLeft >= 600) // Line sensor left hits line

{

lineSensorLeft(); // Perform left line sensor method

delay(70);

moveForward(); // Move forward

}

}

else if(MasterControl == '1') // Xbee reads a '1' from master robot

{

stopMotor(); // Stop vehicle

}

else if(MasterControl == '2') // Xbee reads a '2' from master robot

{

if(counter == 0)

{

delay(5); // While counter is equal to 0, delay(5) then turn left

counter++;

}

else if(counter == 1)

{

counter = 0; // While counter is equal to 1, delay(10) then turn left

delay(10);

}

leftAngle(); // Turn left

count++; // Add 1 to count

}

else if(MasterControl == '3') // Xbee reads a '3' from master robot

{

delay(80);

rightAngle(); //Turn car right

count++;

}

if(distance < maximumRange && distance > minimumRange) //Leader robot too close to follower

{

stopMotor(); // Stop vehicle

delay(25);

}

} //End Loop

//------------------------------------------------------------------------------------------

// Methods to call during loop

//------------------------------------------------------------------------------------------

void moveForward() //Drive forward

{

rightServo.write(85.7); // Set speed of right motor

leftServo.write(84.3); // Set speed of left motor

if(count == 0)

{

delay(100); //Delay after Master turns right before Follower turns right

}

else if(count == 1)

{

delay(40); //Delay after Master turns left before Follower turns left

}

else if(count == 2)

{

delay(65); //Delay after Master turns left before Follower turns left

}

}

void moveBackward() //Drive backwards

{

rightServo.write(105);

leftServo.write(105);

delay(100);

}

void turnRight() //Turn car right

{

rightServo.write(0);

leftServo.write(75);

delay(30);

}

void turnLeft() //Turn car left

{

rightServo.write(75);

leftServo.write(0);

delay(30);

}

void stopMotor() //Stop the car

{

rightServo.write(0);

leftServo.write(0);

delay(10);

}

void lineSensorRight() //Line sensor right turns car left

{

for (angle = 0; angle < 1; angle++) //Tells how long for robot perform task

{

turnLeft();

}

}

void lineSensorLeft() //Line sensor left turns car right

{

for (angle = 0; angle < 1; angle++)

{

turnRight();

}

}

void rightAngle(){

for (angle = 0; angle < 41; angle++)

{

turnRight(); // Perform turn right method

}

delay(30);

}

void leftAngle(){

for (angle = 0; angle < 26; angle++)

{

rightServo.write(75);

leftServo.write(105);

delay(30);

}

delay(30);

}

int ObjectPresent() // Used by ultrasonic sensor to detect objects in path of the robot

{

analogWrite(trigPin,175);

duration = pulseIn(echoPin, HIGH);

return duration / 58.2;

}