# Maze Twinbots

Ly Nguyen, Luke Ireland, Uyen Nguyen

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — The project is designed mainly for the intellectual or experiential gain of the design team members. However, there is also a realization that this project has some inherent flexibility for application such as reconnaissance or search and rescue. The project's purpose is to design two robots that solve a maze in tandem using two separate maze solving algorithms – left wall-following and right wall-following algorithms. The goal of this project is to create both robots to be small, consume low power, be inexpensive, and have a 90% success rate when solving the maze.**

***Index Terms* — autonomous, left wall, maze, right wall, robot**

## I. Introduction

The maze is a 3D perfect maze, meaning it has no loops, no inaccessible areas, and has exactly one solution. The cells are rectangular and intersect at right angles. The start and the exit are separate and located at the outermost wall. Both robots would begin and end at the same locations.

One robot would follow left wall and the other would follow a right wall. As the two robots navigate through the maze, each would store its turns into an array. Once one robot exits the maze, it would analyze the array of turns, represented by characters, and optimize the array to eliminate dead ends. The robot then has a solution to send to the slower robot to exit out the maze.

The robots' key features include power supply, wireless communication, mobility, and autonomy. The power supply system consists of a battery and two regulator subsystems. One regulator regulates 5 V for three sensors and two motors. The other regulates 3.3 V for a microcontroller. Wireless communication between the two robots is implemented by a RF module that transmits and receives characters. Mobility is achieved by the mechanism of wheels and motors whose direction and speed are implemented by pulse width modulation (PWM). Finally, autonomy is accomplished by the integration of software libraries and custom codes which are designed by the team. The final software program is stored in the microcontroller.

## II. System Components

Only major physical components are discussed here. Minor components such as resistors and capacitors are not covered in this section. The electronic system is made up of microcontroller, sensors, servos, regulators, and battery. Mechanical system includes chassis frame, wheels, and rear caster.

### *A. Microcontroller*

The microcontroller is the brain of the entire system; without it instructions cannot be executed, algorithms cannot be implemented, data cannot be read and interpreted and tasks cannot be completed. A Texas Instruments (TI) MSP430F5529 microcontroller was selected over the commonly-used microcontrollers such as Raspberry Pi, BeagleBone Black, BASIC Stamp because it operates at low current (< 100 mA) and voltage (3.6 V maximum) and can be sampled for free. All team members are also familiar using the microcontroller. It has more than enough digital and analog I/O pins for three sensors and two servos. Besides, the microcontroller can interface with the RF module. The microcontroller's memory is 128 KB flash memory which is more than enough to store the maze-solving program.

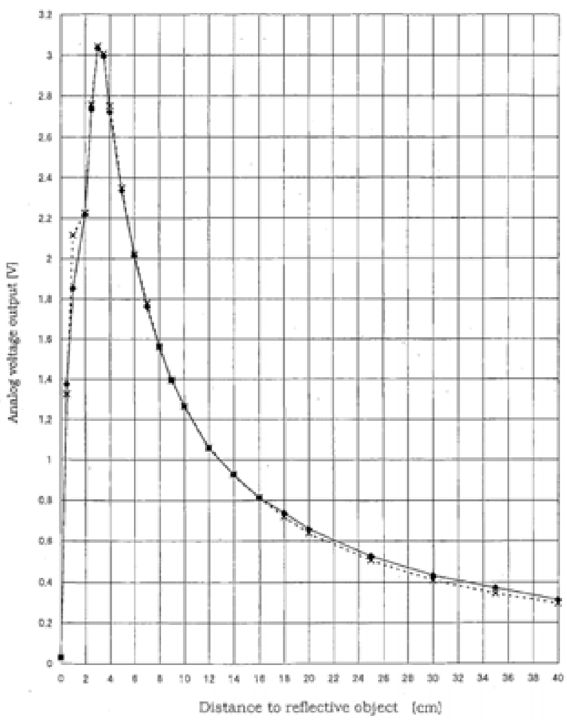
### *B. Sensors*

To perform object detection and avoidance, we use 3 sensors on 3 sides of the robots. The sensor data which is fed into the microcontroller will be converted to distance, in term of centimeters. We have done research on 5 types of sensors. Among these, we chose to test 2 types of sensors: ultrasonic and infrared.

#### *i. Infrared*

Infrared sensors are cheap. They only cost around 10 dollars to 20 dollars. They are easy to set up, have low power consumption and can work quite well with sunlight if we are using Sharp brand. However, after some extensive testing, we come to the conclusion that infrared sensors are not good for our application. The distance is not linearly proportional to the sensor output. As shown in Figure 1, sensor output and distance has a logarithmic relationship so we have to linearize it by generating a best-fit line. As a result, we have to spend more time on sampling data. To avoid this process, we can just use ultrasonic sensors. The range of the infrared sensors are only from 4cm to 30cm. We need longer range since the maze is quite big. Whenever the sensors detect an object that is much farther than 30cm, they start to yield inaccurate results. So we decide to use ultrasonic sensors for our project.

FIGURE 1
IR SENSOR OUTPUT VS. DISTANCE [1]




*ii. Ultrasonic*

Ultrasonic sensors work great even in rain, dusty, or adverse environment. They are not affected by colors and can detect both solid and liquid forms. Their range is from 2cm to 400cm, which is a perfect range for our application. It is close enough to detect walls and far enough to detect opening. Distance can be calculated easily from the speed of sound and the time it takes for the wave to travel back. Like other sensors, they have downsides as well. They cannot detect objects that are too close because the time it takes to travel back is too short for the electronics to respond. If the objects are too far, the sound can grow weak with distance. The density and the material of the objects can distort sensor readings. The sensors consume around 100 milliamps when not in use but it can jump to several amps when used. Despite these disadvantages, they have not caused any problem when we used them. However, after some testing, we discovered that the ultrasonic sensors frequently reset to 0. We solve this problem by retaking sensor reading every time one of the three sensors resets to 0. Another problem is cross interference, which occurs when the signal emitted by one sensor can be read by another sensor. To solve this problem, we face the sensors away from each other. Since they operate on the same frequency, each sensor will take turns reading in the distance, instead of reading at the same time.

*C. Servo*

Motors are required to drive the wheels or any moving parts on the robots. Parallax continuous rotation servo is ideal for our application since it can rotate continuously unlike most standard servos that can only rotate up to 180 degree. However, Parallax standard servos can be easily modified for continuous rotation. The servo does not require an external H bridge or control circuit like stepper and DC motors. The servo's speed is moderate, up to 50 RPM at maximum 6 V which is desirable since sensors have enough time to update their readings and the robots navigate fast enough in the maze. Furthermore, the Parallax servo fits perfectly in the chassis and wheels. It's small and weigh at only 42.5 g. It draws little current (190 Ma maximum at 6 V) when operating in no load conditions.

*D. Regulator*

A LMR61428 step-up switching regulator from Texas Instruments is used to regulate 5 V. It can achieve up to 90 % efficiency and has high switching frequency of up to 2MHz which allows for tiny surface mount inductors and capacitors. A low dropout (LDO) REG113-33 is used to regulate 3.3 V. It doesn't require external capacitors for stability. It also has ultra low dropout voltage of 250 Mv. Both regulators can be sampled for free and available in very small package sizes which are ideal since they take very little space on the PCB. Smaller PCBs cost less and fit better on the robot chassis frame.

*E. Battery*

Four-cell 4.8 V NiMH battery is used to supply power to the sensors, servos, microcontroller, and RF module. The battery is ideal since it is rechargeable, making it suitable for long and extensive use. It's also fairly inexpensive, around $15.00 including a charger. It takes about 2 to 3 hours to fully recharge. Therefore, we don't need to wait too long before we can use the battery again. The battery cells fit nicely in a battery holder that includes a power switch, which is convenient to use.

*F. Chassis and Wheels*

Two Parallax Boe-Bot small robot chassis are acquired from our friends. They are 12.7 cm long and 8.26 cm wide. They are about 4.5 cm tall including the attached

SumoBot wheel that has a diameter of 6.65 cm. The chassis only need two wheels and one rear caster. Its small size fits ideally in a small maze (around 30 cm measured between left and right walls). All electronic components also fit perfectly on it.

## III. Servo Control

The Parallax continuous rotation servos rely on pulse width modulation (PWM) to control the speed and direction of rotation of the servo shaft. However, before PWM can be implemented, the servos' center position needs to be calibrated in order to define the pulse width value at which the servo holds still. The calibration process is often called centering.

### *A. Pulse Width Modulation*

A pulse ranges from 1.3 to 1.7 ms determines the speed and direction of rotation. In order for a smooth rotation, the servo needs a 20 ms pause between the pulses. When the servo's signal pin receives a 1.5 ms pulse width, the servo stops. As the pulse width decreases from 1.5 ms, the servo rotates faster in the clockwise direction. Similarly, as the pulse width increases from 1.5 ms, the servo rotates faster in the counterclockwise direction. As a result, the servo rotates fastest in the counterclockwise direction at 1.7 ms and in the clockwise direction at 1.3 ms.

### *B. Centering*

Manufactured continuous-rotation servos often have trimmer potentiometer which replaces the traditional feedback potentiometer. The trim pot allows the user to calibrate the servo through a screw which can be seen next to the servo's cable. The servo can be easily centered by connecting to a microcontroller which sends a centering signal of 1.5 ms pulse width to the servo. If the servo's shaft rotates, then the servo is not properly centered. Simply insert a Phillips head screwdriver's tip into the hole and gently twist the potentiometer in either direction until it stops turning. [3]

## IV. Wireless Communication Technology

The group has decided to use 915 MHz RF communication. This will be implemented using the Anaren AIR Module which is an integrated transceiver and antenna that interfaces to the microcontroller via SPI.

The team could have very well used Bluetooth technology to accomplish the same task since the range that the robots will communicate is well within the range of Bluetooth however, the team felt that it could be of use to become familiar with an increasingly popular form of wireless communication being used in industrial applications today.

The Anaren AIR module was also chosen because it allowed the team to have access to the CC110L BoosterPack for the MSP430F5529 LaunchPad which made it much easier to prototype the design. The AIR module also has the added advantage of being a very low consumer of power once again aligning with the design constraints of the project; it consumes just 15 mA in active receive mode and just 200 nA in sleep mode. It can be powered at the same voltage as the microcontroller which eases PCB design. [2]

The Anaren AIR Module also has very convenient packet handling and data transmission features that assist in accelerating the design of our system as well as the power efficiency. The AIR module has transmit and receive data buffers that can hold up to 60 bytes of data at a time, this means that in one transmission the AIR module can send 60 bytes of data, corresponding to 60 turns in our application, greatly reducing the current drain of the transceiver since only one transmission will be required for most maze configurations.

## V. Programming The Microcontroller

One of the major issues encountered with this project was being able to load a program onto the microcontroller on the PCB. Originally, the team attempted to use Spy-Bi-Wire to program the microcontroller since it only requires four pins (Vcc, Ground, RST, and TEST) and can also use the emulator from a LaunchPad which lowers cost however, we soon found that this method of programming is highly sensitive and difficult to implement. The first PCB the team designed attempted to use Spy-Bi-Wire but the team was unable to connect to the programmer and could therefore not load a program. While attempting to debug the original board it was discovered that Spy-Bi-Wire is extremely sensitive to board capacitances, line capacitance, and other noise sources since its two wire data is converted to four wire internally using control logic, this requires very accurate timing that can easily be disrupted. Our design did not consider such problems and did not design to mitigate these problems. After more research into programming methods for the microcontroller the team decided to switch to a JTAG interface which requires more pins but is much more reliable and faster than Spy-Bi-Wire. The second revision of the PCB design utilized the JTAG interface and was able to successfully load programs to the microcontroller. We were able to secure a JTAG Flash Emulation Tool (FET) from the TI lab which eliminated the increased cost that paying retail for a FET would bring.

## VI. Maze Construction

The maze(s) that the robots will be solving is (are) an important portion of the design constraints for the project. Since both robots are utilizing wall following algorithms to navigate through the maze, the maze must adhere to the limitations of this algorithm; that means that the solution cannot be inside the maze, it must be on the outer edge. The maze will also be restricted to having one unique "best" solution which both robots will be able to, ideally, realize after optimizing their respective paths through the maze. The maze must be made of a material that will adequately reflect the ultrasonic sensors pings and must also be tall enough such that pings actually bounce off the maze walls as opposed to traveling over them as well as be wide enough for the robots to navigate through and obtain clean sensor data. The maze must also be reconfigurable to allow for proper testing and proof of concept. The easiest and chosen material for this maze will be 2" X 8" wood cut in to 1 foot lengths since it can stand on its side and be easily reconfigured.

## VII. Hardware Design

Figure 2 is a block diagram of the whole system of each robot. One battery pack is used to power all the electronics. Since the sensors can only tolerate input voltage of 5 V and the servos' power input ranges from 4 to 6 V, a 5 V regulator is needed to boost the 4.8 V battery to a 5 V. The microcontroller and RF module operate typically at 3.3 V. As a result, a 3.3 V regulator is used to lower 4.8 V to 3.3 V. Switching regulator is favored over linear regulator because it has a much higher power efficiency, typically 85 % compared to linear regulator's typical efficiency of 40 %. Even the new linear low drop-out (LDO) regulators are not as power efficient as the switching regulator. Since linear regulator's efficiency is low, it generates a lot of wasted energy in the form of heat which has to be dissipated by heat sink. In addition, it's difficult to drive loads over 200 mA and reduces battery life. Switching regulator's efficiency doesn't depend on input voltage as much as linear regulator. However, switching regulator generally is more complicated to set up than linear regulator because it has more pins and requires more external components.

A LDO regulator is used to regulate 3.3 V instead of a switching regulator since the RF module and microcontroller consume low power. Both the microcontroller and RF module draw very little current, less than 100 mA and 14.7 mA respectively. The REG113-33 LDO regulator can output up to 400 mA which is more than enough for the RF module and microcontroller.

Each ultrasonic sensor consumes 15 mA. Three sensors would consume 45 mA. The maximum current draw for each servo is 190 mA. Therefore, two servos would draw at most 380 mA. The three sensors and two servos draw a maximum of 425 mA. The LMR61428 switching regulator can provide up to 2.85 A which is more than enough for the sensors and servos.

Table 1 summarizes the voltage and current required by each component. All the listed components can draw up to around 540 mA. The battery can handle much more than that. It

TABLE 1

COMPONENTS' ELECTRICAL CHARACTERISTICS

| **Component** | **Voltage (V)** | **Current (mA)** | **Qty** |
|---|---|---|---|
| Servo | 4 – 6 | 15 - 190 | 2 |
| Ultrasonic sensor | 5 | 15 | 3 |
| Microcontroller | 1.8 – 3.6 | < 100 | 1 |
| RF module | 1.8 to 3.6 | 14.7 | 1 |
| | Total | < 539.7 | 6 |

The sensor data are sent to the microcontroller to be interpreted. The sensors' analog data is converted into a measurement that can be understood by humans such as centimeter or inch by the software program. The microcontroller is responsible to send out pulse widths to control the wheels' movement based on the sensors' measurement. Once the robot finds the exit, it sends an array of characters that represent the turns it takes while navigating the maze from the microcontroller to the RF module via the SPI interface which is bidirectional. Then the RF module sends the characters to the second robot's RF module which in turn sends the characters to the second robot's microcontroller to be interpreted.

FIGURE 2

SYSTEM BLOCK DIAGRAM

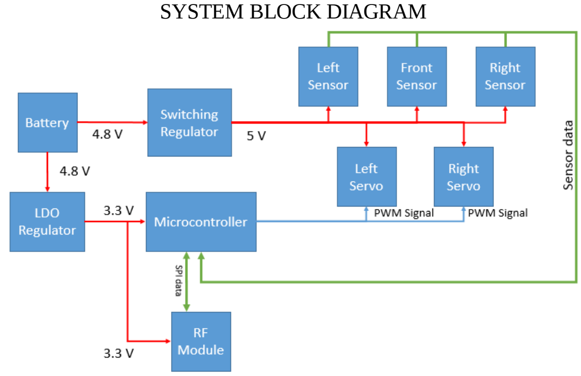


Figure 4 shows the schematic of the REG113-33 and LMR61428 regulator circuits. The capacitors help to filter out noises coming in and out of the regulators. RF1 and RF2 determine the output voltage. Therefore, changing RF1 and RF2 to different values result in different output. RF1 and RF2 are related in (1). A value of 150kΩ is suggested for RF1. RF2 can then be calculated from the equation given a 5 V output. VOUT is the output at pin 7 of LMR61428.

$$RF2 = RF1 /[(VOUT/ 1.24) - 1] \quad (1)$$

CF1 feeds back most of the AC ripple at VOUT to the FB pin. RFQ sets the frequency for the regulator which can operate between 300kHz and 2MHz. CI, CNR, and CO are not required for stability. LED1 and LED2 serve as status indicators. If the regulators regulate voltage, then both LEDs must be on. If not, then they are off.

Figure 3 shows the microcontroller interfacing to the RF module. The headers JP3, JP4, JP9, and JP12 are used for IR sensors. Headers JP5 and JP8 are used to connect to two servos. Headers JP16, JP6, and JP13 are used to connect to ultrasonic sensors. The JTAG header pins are connected to a MSP-FET (flash emulation tool) which is used to program the MSP430F5529 chip.

FIGURE 3

MICROCONTROLLER INTERFACE

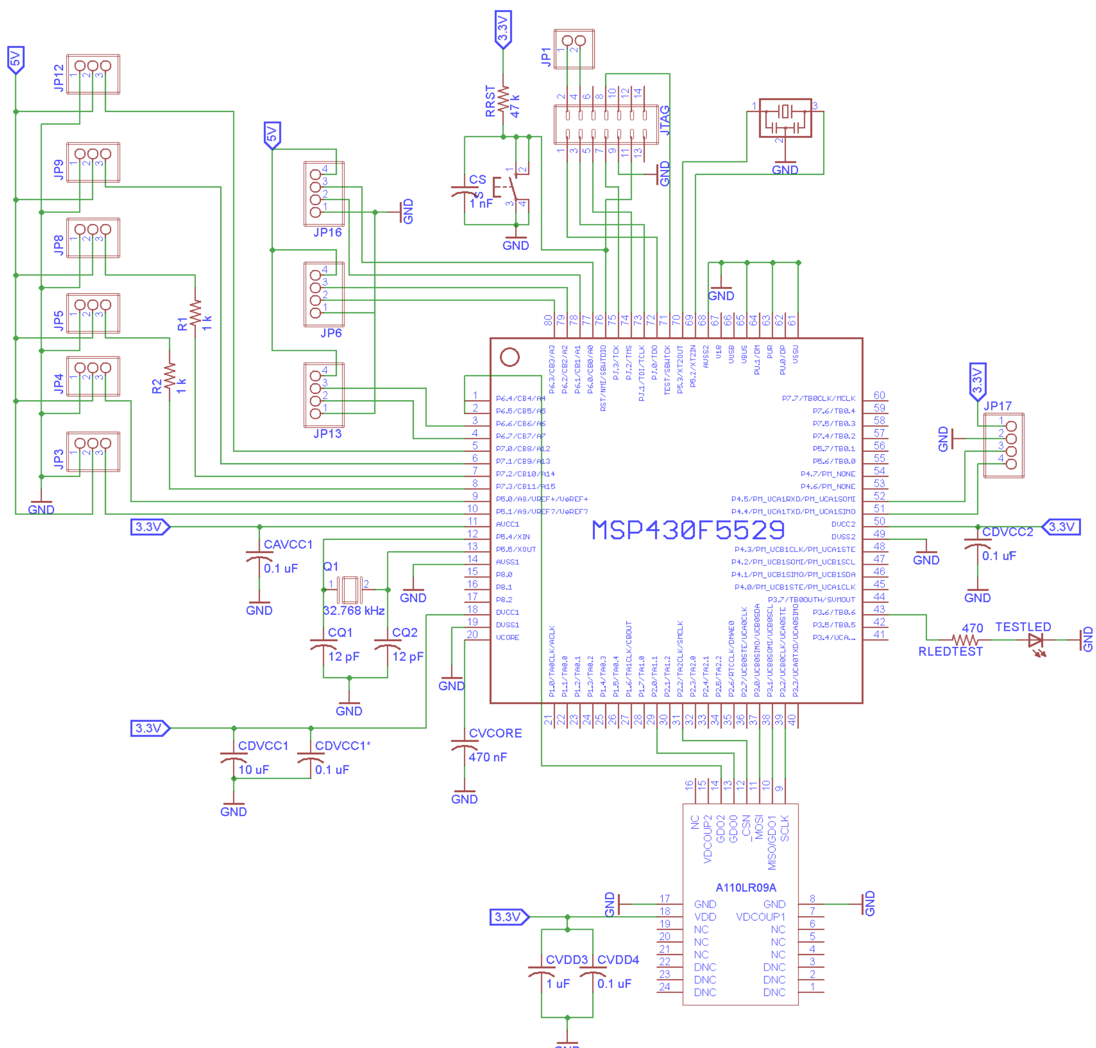


FIGURE 4

POWER SUPPLY

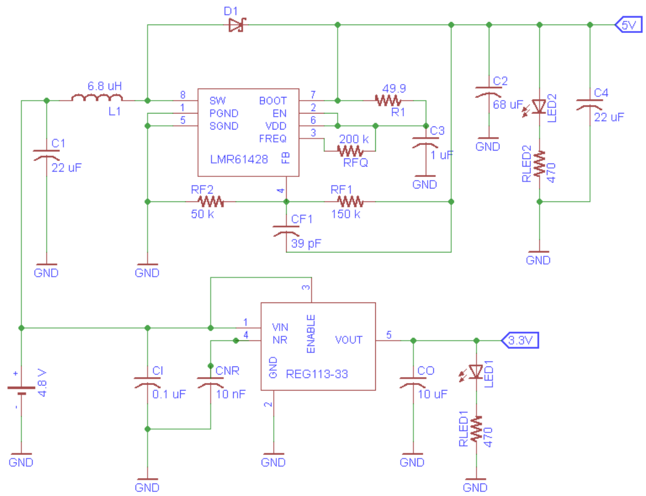
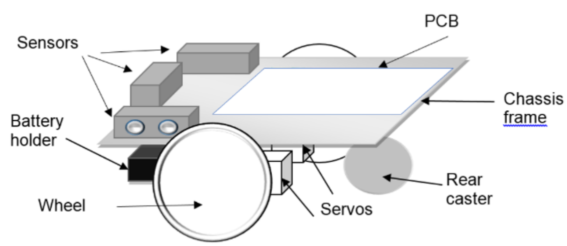



Figure 5 shows the robot model. It's best to keep the electronics further away from the motors which might cause electronic noise interference. Heavy components such as battery and servos are placed at the bottom of the chassis. Lighter electronics are placed on the top away from the motors. The PCB is placed as close to the sensors as possible so that the sensors can connect to the header pins on the PCB. The ultrasonic sensors should be placed as high as possible on the robot in order to correctly detect obstacles. The sensors can also be placed vertically instead of horizontally as shown in the Figure 5.

FIGURE 5

ROBOT MODEL




## VIII. Wall Follower

The algorithm used to solve the maze is called wall follower, also known as left-hand rule or right-hand rule. It is equivalent to a person placing one hand on the wall at all time and walks through the maze until he reaches the exit. This technique is also called the LSRB algorithm. Every time the robot turns left, it stores an L in an array. If it goes straight, it stores an S. If it turns right, it stores an R and if it turns around, it stores a B. The robot which uses the left-hand rule follows 4 conditions.

1) It should turn left whenever it can.
2) It should go straight if it cannot turn left.
3) It should turn right if it can neither turn left nor go straight.
4) It should turn around if the previous 3 conditions are not met, which means it has encountered a dead end.

Likewise, the robot which uses the right-hand rule follows 4 similar conditions.

1) It should turn right whenever it can.
2) It should go straight if it cannot turn right.
3) It should turn left if it can neither turn right nor go straight.
4) It should turn around if the previous 3 conditions are not met, which means it has encountered a dead end.

We choose this method because it is fast and requires little memory. Also, the robot will always find the exit if there is one. The downside to this is that the robot will keep going in a circle if the start or the end points are inside the maze. To solve this problem, we are going to put the entrance and the exit on the outer wall. Also, it may not offer the shortest way out if the maze has multiple solutions. Fortunately, our maze only has one solution so this should not pose any problem.

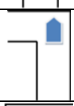
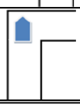
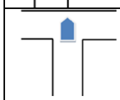
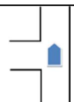
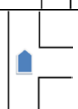
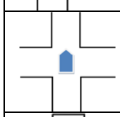
## IX. Maze Solving Algorithm

We have 2 robots so one of them is going to use the left-hand rule. The other will use the right-hand rule. Initially, they follow the same path. At some point, they'll take different route at a junction. The robot that finds the exit first will send signal to the other robot, telling it to stop searching and exiting the maze using the solution. While in the maze, the robots will have to make 6 decisions:

1) Look for the next junction
2) Identify the type of the junction
3) Make a turn
4) Store a character in the array
5) Look for the exit
6) Listen for incoming signal to see if the other robot sends the solution

The robots encounter 9 possible types of passages, as shown in Table 2. When the robots enter the maze the first time, they don't know the layout of the maze, so it's inevitable that the robots will encounter dead ends. However, when the robots enter the maze the second time, they should avoid all dead ends and only follow the path that leads to the exit. Every time the robots encounter a dead end, it stores a B in the array. So to eliminate the dead ends, we have to get rid of all the Bs. To get out of a dead end, we replace a three-letter sequence by another letter, as shown in Table 3. The sequence always have a B as the second letter, so when we simplify this sequence, we are going to search for Bs.

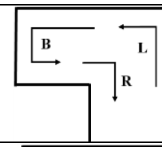
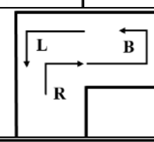
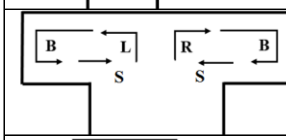
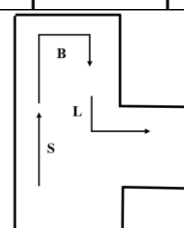
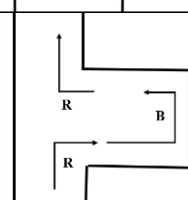
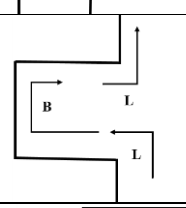
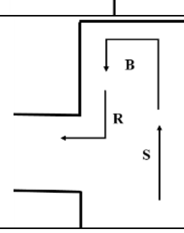
TABLE 2
POSSIBLE PASSAGES IN A MAZE

| Passages | Left-hand Robot | Right-hand Robot |
|---|---|---|
|  | Go straight and search for the next junction | |
|  | Turn left | |
|  | Turn right | |
|  | Turn left | Turn right |
|  | Turn left | Go straight |
|  | Go straight | Turn right |
|  | Turn left | Turn right |
|  | Turn around | |

At the beginning of the program, we create 2 arrays. One is used to store the current path of the robot. Let's call it the path array. The other is used to store the solution. Let's call it the solution array. The robot will navigate through the maze using the left-hand rule or the right-hand rule, as long as it doesn't detect an exit and it doesn't receive the solution from the other robot. If it exits the maze, it'll stop and scan through the path array to look for Bs. It'll replace the three-letter sequence in Table 3 with another letter until all Bs disappear from the path array. For an example, supposed the robot has this string stored in the path array: LRBRSSBLL. We have 2 sequences that we need to replace: RBR and SBL. RBR is replaced by an S and SBL is replaced by an R. After optimization, the array is reduced to LSSRL. There's no B in the array, which means the next time the robot enters the maze, it won't encounter any dead ends. Now the path array becomes the solution.

It sends this solution to the slower robot which is still in the maze. The slower robot stores the solution in the solution array. Then it optimizes its own path array and compares it against the solution. If the path array doesn't match with the first portion of the solution, the slower robot will continue to navigate through the maze and keep optimizing its own array until it matches with the solution.

TABLE 3
THREE-LETTER SEQUENCE SUBSTITUTION

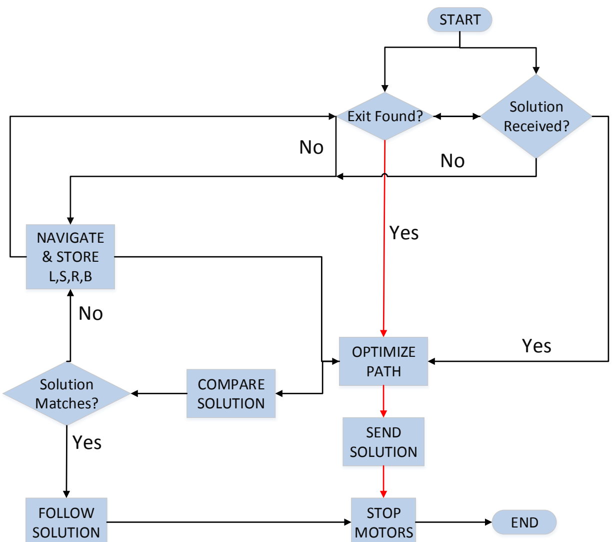
| Junction Type | Sequence of Turns | Substituting Letter |
|---|---|---|
|  | LBR | B |
|  | RBL | B |
|  | LBS | R |
| | RBS | L |
|  | SBL | R |
|  | RBR | S |
|  | LBL | S |
|  | SBR | L |

For an example, supposed the slower robot reduces its path array to LS. It compares this sequence to the solution LSSRL. It sees that the solution also starts with LS. So it uses SRL, the rest of the solution, to get out of the maze.

Every time the left or the right sensor detects an opening, the robot knows that it has encountered a junction. It scans the solution and determine what kind of turn it needs to do. After LS is an S, so at the next junction, it will go straight. After that is an R, so at the second junction, it will turn right. After that is an L, so at the last junction, it will turn left to exit the maze. Figure 6 shows the flow chart of the algorithm.

FIGURE 6

SOFTWARE INTEGRATION




## X. Conclusion

In conclusion, this project seeks to build a team of two maze solving robots that will utilize two different maze solving algorithms to search for the most efficient route through the maze. This project incorporates many important fundamental skills that are required of electrical engineers including: PCB design, system integration, wireless communication, and embedded system programming. The team has expanded our knowledge greatly in the details pertaining to the fields mentioned above such as sensor technology, various wireless communication technologies In addition to these skills the team has learned how to build up a system from the ground up by utilizing available reference designs as well as product datasheets to meet any and all design constraints that pertain to our system. The team has developed critical soft skills such as project management, delegation, and team communication. Overall, this project will serve as a solid introduction to the fundamentals of beginning, designing, and building a project from start to finish and will provide the engineers with valuable skills that they can carry through the rest of their professional lives.

## Biography

Luke Ireland is currently a senior at the University of Central Florida and will be graduating with his B.S.E.E. in the top 5 percent (Magna Cum Laude) of his class in May 2015. Luke has accepted a position as a Test Engineer at Intersil Corporation in Palm Bay, Florida beginning May 11, 2015. He plans to take one year off and then return to earn his M.S.E.E specializing in semiconductor electronics from UCF as a part time student.

Ly Nguyen is currently a senior at University of Central Florida and will be graduating in May 2015. She plans to pursue master degree in robotics while working for NASA at Kennedy Space Center as an electrical engineer.

Uyen Nguyen is currently a senior at University of Central Florida and will be graduating in May 2015. She plans to pursue master degree in analog or digital circuit design.

## References


[1] Sharp Microelectronics, "Analog Output Type Distance Measuring Sensor," 5 March 2007. [Online]. Available: http://media.digikey.com/pdf/Data%20Sheets/Sharp%20PDFs/GP2Y0A41SK0F_Spec.pdf. [Accessed 25 October 2014].

[2] Texas Instruments, "CC110L," Texas Instruments, January 2014. [Online]. Available:http://www.ti.com/product/cc110l?keyMatch=cc110L&tisearch=SearchEN. [Accessed 10 October 2014].

[3] A. Lindsay, "Activity 5: Centering the Servos," 29 February 2012. [Online]. Available: http://learn.parallax.com/node/185. [Accessed 11 September 2014].