2018 IEEE SoutheastCon Hardware Competition

Avast, Me Hearties!

Brittani Carroll, Electrical Engineering

Brooks Brown, Electrical Engineering

Project Advisor: Mr. Mark Randall

April 26, 2018

Evansville, Indiana
Acknowledgements

The 2018 IEEE SoutheastCon Hardware Competition team would like to thank the University of Evansville’s EECS Department for funding the project and the Academic Funding Board for funding our travel. We would also like to thank and acknowledge Mr. Mark Randall (project advisor), Dr. Christina Howe, Dr. Marc Mitchell, Dr. Mohsen Lotfalian, and Vicky Hasenour for support and technical assistance in the project.
Table of Contents

I. Introduction

II. Statement of Problem

III. Project Design
   A. Hardware Design
   B. Software Design
   C. Standards and Constraints

IV. Costs

V. Results

VI. Conclusion and Recommendations

VII. References

VIII. Appendices
   A. Appendix A: Playing Rules
   B. Appendix B: Source Code

Table of Figures

1. Overall view of map
2. Figure of track

3. Picture of our logo on the flag and on the robot

4. Images of parts selected

5. Picture of final robot

6. Picture of test track

7. PCB schematic

8. PCB design

9. Hardware flow-chart

10. Software flow-chart

Table of Tables

1. Breakdown of tasks and points for each round

2. Breakdown of budget from EECS
I. Introduction

The 2018 IEEE Southeast Con Hardware competition took place at the University of South Florida and was a pirate theme. The competition consisted of six different tasks to be completed that were each worth different amounts of points. The tasks were as follows: read the treasure map, lower the gangplank, obtain the treasure chest key, raise a flag, pick up the treasure chest, and then lastly return to the ship and raise the gangplank to end the round. The robot had many specifications that it had to meet in order to be able to compete in the competition. Each task was worth a certain amount of points and the overall goal was to obtain the most points in the least amount of time [1].

II. Statement of Problem

The IEEE Southeast Con Hardware competition is an event for college students in the southeast United States to participate in. For this competition, students design and build a robot that follows the course and completes the tasks given for each year. The competition takes place at a different university in the southeast every spring, and this year it was held at the University of South Florida in Tampa, Florida [1].

Every year the competition also has a different theme and course, and this year the competition was pirate themed with the course being an island that had a designated treasure map. Figure 1 shows the overall view of the map and Figure 2 shows an overview of how the track was laid out [1].
Figure 1: Overall view of the map

Figure 2: Image of track
The robot was designed to follow the given treasure map and complete the assigned tasks in under four minutes. For each task that was completed in the round, the team was rewarded a certain amount of points that was outlined in the rules with the overall goal being to earn the most amount of points in the least amount of time. An overall breakdown of tasks, and the points designated with each task, can be seen in Table 1.

**Table 1: Breakdown of tasks and points for each round [1]**

<table>
<thead>
<tr>
<th>Actions</th>
<th>Points</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remaining time</td>
<td>0 to 240</td>
<td>A point was issued for every second that the robot completed the round early. Formula: Points earned = 240-Completion Time (sec)</td>
</tr>
<tr>
<td>Display code correctly</td>
<td>200</td>
<td>Robot’s LCD Screen displayed correct binary signal received and the corresponding decimal code/route number.</td>
</tr>
<tr>
<td>Activate Correct Destination A (Lower gangplank)</td>
<td>50</td>
<td>Robot maneuvered to and pressed correct button at destination A to lower the gangplank.</td>
</tr>
<tr>
<td>Cross gangplank</td>
<td>100</td>
<td>Robot crossed the bridge while indicator LED was green (or directly after pushing correct button) and did not touch the water.</td>
</tr>
<tr>
<td>Cross gangplank without active Destination A</td>
<td>50</td>
<td>Robot crossed the bridge while indicator LED was red (when Destination A was not activated).</td>
</tr>
<tr>
<td>Activate Destination B (Obtain treasure chest key)</td>
<td>200</td>
<td>Robot maneuvered onto the correct pressure pad (could not proceed to treasure chest without completing).</td>
</tr>
<tr>
<td>Pick up and store treasure chest</td>
<td>200</td>
<td>Robot picked up and stored the treasure chest in or on robot successfully after completing Destination B for at least five seconds (could still get points if the chest is dropped).</td>
</tr>
<tr>
<td>Move treasure chest</td>
<td>100</td>
<td>Robot moved treasure chest out of designated area but did not store it (not eligible for points if the robot picked up and stored the treasure chest).</td>
</tr>
<tr>
<td>Fully raise flag to top</td>
<td>200</td>
<td>Robot had to turn the captain’s wheel clockwise in order to raise the flag to the top of the flagpole. Points were rewarded if the final flag position reaches the top of the flagpole (4.75 to 5.25 turns).</td>
</tr>
</tbody>
</table>
The first task was to read the treasure map. This task was done by reading an infrared (IR) signal in an eight-bit message, whose last three bits revealed one of eight different routes. If the robot read and displayed the correct code and route number onto an LCD screen, then 200 points were rewarded. The next task was to lower the gangplank and was completed by pressing a limit switch at the destination determined in the first task. Once the correct limit switch was pressed 50 points were rewarded and the robot was cleared to cross the gangplank. When the robot crossed the gangplank onto the island without “falling” into the water, the team received another 100 points. If the incorrect button was pushed and the robot still crossed the gangplank, then the team only received 50 points [1].

Next, the robot had to obtain the key to the treasure chest by depressing a lever to activate a limit switch. Once this happened, a green LED confirmed successful pad activation and the team earned 200 points. If the incorrect pressure pad was activated, the team received no points. After obtaining the key to the treasure chest, the robot had raise the flag to claim their spot on the
island, which was worth a maximum of 200 points. In order to raise the flag, the robot had to rotate a wheel in the clockwise direction. If the wheel was rotated three times, the team received 100 points and the RGB LED built into the rotary encoder would turn from red to yellow. Five rotations resulted in the full 220 points and turned the LED green. If the wheel was turned more than five and three fourths times, the LED would turn red again and the team received no points [1].

The robot also had to pick up and store the treasure chest and bring it onto the ship. If the robot did not successfully pick up the treasure chest but moved it outside of an orange outline, the team got 100 points added to their score. If the treasure chest was successfully picked up and stored on the robot for at least five seconds, then the team received another 200 points but not the 100 for moving the treasure chest. Activation of Destination C, pressing a limit switch to end the round, was the next step and was worth 100 points. Lastly, returning to the correct final destination on the ship, with the treasure chest, after activation of Destination B, and pressing the limit switch to signal the end of the round was worth 300 points. Once the button to raise the gangplank was pushed the round ends and the team was no longer able receive points [1].

Although completing tasks got the teams the majority of the points, there were other ways to earn points too. The first way was to complete the round quickly. The team had to complete the round in under 4 minutes and earned a point for every second they finished early. An extra 150 points each could have also been earned if the team created and wore a team shirt, as well as created a team flag, and lastly a logo which is shown in Figure 3 [1].

The robot had to also meet specific requirements in order to compete. The first major requirement was that the robot had to be autonomous, meaning that it could not have any wired or wireless communication, other than the IR receiver. It also had to contain the school logo and
not exceed 12”x12”x12” at the beginning of the round. During the round, it was able to extend but still could not exceed 20”x20”x20”. The robot could not split into multiple modules and had to remain singular, as well as the robot could not have the ability to fly [1].

The competition began with two qualifying rounds that were both scored by the same criteria. After the qualifying rounds, the top four teams with the best scores continued onto the semi-finals and elimination round. The teams ran their robots through another course and the top two of the four teams proceeded onto the finals, where the teams again competed and the highest scoring team was proclaimed the winner [1].

In order to graduate, a number of requirements had to be met including building a robot that completed the following tasks:

- Read the IR signal to retrieve the correct treasure map
- Activate Destination A
- Traverse the gangplank successfully
- Activate Destination B
- Maneuver to the treasure chest
Figure 3: Picture of our logo on the flag and on the robot
III. Project Design

A. Hardware Design

In order to accomplish the tasks laid out in the problem definition a complete design was required, with the first being for navigation. To achieve the correct speed, motors were selected to match the speed desired in RPM. The motors selected were the Pololu 12V 131:1 Metal Gear Motor with Encoder as seen in Figure 4a [2]. The wheels chosen were the Andy Mark 4” Mecanum Wheel Set (2x Left, 2x Right) as seen in Figure 4b [3].

In order to stay inside the size requirements, the bottom base of the chassis had to be 11” x 7” before attaching the wheels to the outer edges. Because of this restriction, the motors had to be small enough to fit under the width of the chassis. The motors chosen also had encoders that could accurately count the number of times the wheel rotated. To allow for easier access to the motors with a microcontroller, motor controllers were used. The controllers chosen were the Cryton 10A 5-25V Dual Channel DC Motor Driver as seen in Figure 4c [4]. The motor controllers selected included test buttons, were PWM controlled, and had a 10A limit which was twice the stall current of our motors. The distance sensors used were "time of flight" sensors called Adafruit VL53L0X Time of Flight Sensor as seen in Figure 4d [5]. These specific sensors where chosen due to their accuracy, range of sensitivity, small size, and the cone of sensitivity.

The first task to accomplish was to decode the IR signal in order to figure out which route to follow. To do this a hole was cut in the middle of the chassis and an IR receiver diode, shown in Figure 4e, was placed in it to read the blinking IR signal [6]. Then a microcontroller was used in order to receive and decode the signal. The microcontroller selected was the Arm Discovery Board STM32F407, which is shown in Figure 4f [7]. After the IR signal was received and the
robot knew which route to take, the next task was to maneuver to the correct Destination A and press the limit switch. The robot had mecanum wheels so it could strafe certain directions rather than having to turn. Once the robot reached Destination A, there was a linear actuator that extended and pressed the limit switch to activate Destination A. The linear actuators chosen were the L12 12V 100mm 100:1 PLC/RC Miniature Linear Actuator shown in Figure 4g [8]. After the limit switch was pressed, the robot crossed the gangplank. The wheels were four inches and will help with this task because they were able to grip the ramp best.

At the bottom of the ramp, the robot had to activate Destination B by activating a pressure pad that was a wooden board with a limit switch. The 4-inch wheels were large enough to press the board down and activate the limit switch. After activating Destination B, the next step was to raise the flag. To accomplish this task, an attachment was 3D printed and inserted onto a motor, equipped with encoders. The motor spun the rotary encoder the correct distance and direction to raise the flag. The motor used for this task was the same motor used for navigation. Then the robot was programmed to pick up and store the two-pound treasure chest. In order to do successfully do this, the robot had permanent magnets attached to the end of an arm and a servo to raise and lower the arm. The servo selected was the REV Smart Robot Servo shown in Figure 4h [9]. Once the box was picked up, the robot carried it back onto the ship and used the linear actuator to press the limit switch again, which ended the round. An overall picture of the final robot is shown in Figure 5.

To test the design, a prototype of the navigation which included the chassis, motors, motor controllers, and distance sensors were created, and the navigation functions were programmed and tested. As this was happening, the final design was finalized, and a prototype of the full competition robot was built and programmed to complete the entire round on the test
track in the lab shown in Figure 6. Programming the prototype pointed out what parts of the design needed to be altered, allowing the PCB board and final robot to be designed and assembled. Our PCB schematic and design are shown in Figure 7 and 8. Also, a flow-chart of the hardware design is shown in Figure 9. Once the final robot was assembled, the code and robot were finalized to work as efficiently as possible for the competition.

Figure 4: Images of the parts selected (a) Pololu 12V 131:1 Metal Gear Motor with Encoder, (b) Andy Mark 4” Mecanum Wheel Set, (c) Cryton 10A 5-25V Dual Channel DC Motor Driver, (d)
Adafruit VL53L0X Time of Flight Sensor, (e) IR receiver diode, (f) Arm Discovery Board STM32F407, (g) L12 12V 100mm 100:1 PLC/RC Miniature Linear Actuator, (h) REV Smart Robot Servo

**Figure 5:** Picture of final robot

**Figure 6:** Picture of test track
Figure 7: PCB schematic

Figure 8: PCB design
B. Software Design

Although a majority of the software was done in Keil µVision and programmed on the Arm Discovery Board STM32F407 in the C programming language, the distance sensors were programmed in C using an API for the LPC1768. The LPC1768 and Arm Discovery Board communicated via UART to send and receive distances found. The code consisted of several different functions to complete each task assigned. The flow of the functions called in the code consisted of the layout in Figure 10.

Figure 9: Hardware flow-chart

Figure 10: Software flow-chart
The first function called was reading the IR signal. In this function there was an input port that displayed the signal the infrared receiver, which was attached to the bottom of the robot, was reading. First, it looked for the start burst of the signal being sent. Once the start burst was found, the code looked to see if all ones were being sent, which represented the positioning signal, or if five logical zeros and then three randomly generated bits were being sent which represented the route. This loop kept going until a route was received.

Once a route was received the next function the code called was the correct Destination A function, depending on the route number. In the Destination A function, a turn left or turn right function, which turned the robot the designated direction by counting encoder ticks in an external interrupt, was called to turn the robot whichever direction was needed. Then a forward encoder function was called to move the robot forward a certain number of encoder ticks. The opposite turn function was then called to turn the robot back forward so the robot could level using the level back function, which read the value of the distance sensors on the back and rotated the robot the needed direction until the distances of both sensors were equal. After leveling back, the robot moved forward using a function that moved the robot forward until the back sensors were a distance away from the wall to ensure that the side sensor was able to ping off of the board the Destination A button was placed on. Next, the robot strafed sideways until the distance sensors read the sent value and was close enough to push the button. After extending the linear actuator, utilizing pulse width modulation, the robot turned back towards the center, moved encoder ticks forward, turned back forward, leveled, and then utilized encoder ticks to go down the ramp.

After completing Destination, A, the designated Destination B function was called. In this function, the robot called the turn function to turn the robot the correct direction. It then
utilized encoder ticks to move forward and activate Destination B and then go backwards. Once
off Destination B, the robot turned back forward and moved forward towards the end of the track
with encoder ticks. Once close enough to the wall, the robot moved forward by reading the front
sensor and stopped once reaching a certain distance away from the wall. It then again called the
function to turn the robot the needed direction and move it forward with encoder ticks until the
robot was nearly centered. Once here, the robot pinged the top sensors in the front and moved
either left or right until both sensors read a distance of less than 100mm which indicated they
were lined up with the flag.

Once the robot was lined up with the flag, the code called the flag and chest function.
The first thing in this function was to extend the track actuator, utilizing pulse width modulation,
and then spin the motor, again utilizing pulse width modulation, and finally to bring the track
actuator back in. The robot then moved backward and right with encoder ticks a small amount to
get to the treasure chest. At the treasure chest, a function was called to bring the arm that would
pick up the treasure chest down and then back up utilizing the servo. In this function, the pulse
width modulation value of the servo increased and then decreased to bring the arm down and
then up. Finally, the robot moved backwards and back onto the ship utilizing the encoders and
distance sensors. Once on the ship, the function for Destination C was called. This function was
very similar and practically the same as the function for Destination A.

Throughout the whole round, there were also level functions for each side of the robot
and a center function. These functions allowed for the robot to straighten itself out and make
sure it was always in the correct position. Every time the robot moved towards a wall and
stopped, a level or center function was used. The use of these functions helped keep the position
of the robot more accurate, so it was better able to complete each given task.
C. Standards and Constraints

Since the robot was powered off Li-Po batteries, there was an environmental and safety concern if they would catastrophically fail. However, this was very unlikely to occur so therefore it was not a high concern. Another major safety concern was the magnet being used to pick up the treasure chest. If the magnet was too strong, it could have cleared the memory on our microcontroller as well as disrupt pacemakers and other medical devices. To ensure this did not happen, the team thoroughly researched the strength of magnets before ordering. There was also the concern of safety since our robot was not water proof. If a spill occurred, then it could have destroyed the robot and risked injury to any personnel near. Lastly, the design remained simple to create better manufacturability and, therefore, a safer robot.

While preparing and participating in the 2018 IEEE Southeast Con Hardware competition, it was important to consider the IEEE code of ethics [10]. The most important ethic that the team followed was “to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others” by participating in group discussion between teams on a messaging app in order to discuss errors, concerns, and questions regarding the rules and the competition. By doing so and responding correctly, the team also followed the ethic “to avoid injuring others, their property, reputation, or employment by false or malicious action” [10]. Throughout the project, the team also properly cited sources and gave credit to those who deserve it.

IV. Costs

Table 2 shows the breakdown of costs for the project. The team received $2,950.00 from the EECS department to purchase parts to build the track, robot, and necessary pieces for the
design portion of the competition. The track materials cost $282.88 while the robot materials cost $1,887.14. The shirts and flag for the design portion totaled to $70. In the end, the team ended up using a total of $2,240.02, which left $709.98 remaining in the budget.

<table>
<thead>
<tr>
<th>Category</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot Materials</td>
<td>$1,887.14</td>
</tr>
<tr>
<td>Track Materials</td>
<td>$282.88</td>
</tr>
<tr>
<td>Design Competition</td>
<td>$70.00</td>
</tr>
<tr>
<td>Received from EECS</td>
<td>$2,950.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$2,240.02</strong></td>
</tr>
<tr>
<td>Remaining</td>
<td>$709.98</td>
</tr>
</tbody>
</table>

The team was also able to receive $2,946.36 from AFB for travel costs. These costs included registration, flight, baggage, hotel, and the rental car as well as everything associated with it.

V. Results

Overall, the main purpose of this project was to build a robot that was capable of competing in the 2018 IEEE Southeast Con Hardware Competition at the University of South Florida in Tampa, Florida in April. The robot build was able to compete and finished 9th place at competition out of over 50 teams. The average time it completed the course was just under two minutes while the fastest time was one minute and thirty-seven seconds.
VI. Conclusion and Recommendations

In the end, the robot was able to successfully navigate the course and competed and placed well at the 2018 IEEE Southeast Con Hardware Competition. If this robot would be built again, the team would recommend using wheels that are not mecanum. The main purpose of the mecanum wheels was to strafe sideways but when weight was added to the robot and not distributed equally the robot would not strafe well and would shift causing it to actually go diagonal. The team would also recommend utilizing different distance sensors with a narrower cone or testing the distance sensors on the track when no walls were surrounding. This is recommended because sometimes the distance sensors would ping over the wall or onto the floor which caused several problems at competition. Other than these minor changes, though, which were able to be fixed by altering code the team would recommend rebuilding the robot utilizing the same design.
References


Appendices

Appendix A: Playing Rules

IEEE Southeastcon 2018
IEEE USF: Hardware Competition Rules
**Manifest:**

Terminology 2

Avast, Me Hearties! 2

Overview 2

The Treasure Map 2

Figure 1: SoutheastCon18 Treasure Map 3

The Route 3

Table 1: Coordinate Locations on Treasure Map 5

Figure 2: SoutheastCon18 Playing Field 6

Playing Rules 7

Doubloon earning locations 9

Doubloons earned outside of round 9

Robot Specifications 9

Team Logo Specifications 9

Team Flag Specifications 9

Field Specifications 10

Figure 3: Playing Field Paint Colors 11

Initial Signal 11

Figure 4: IR Signal Spec Description 12

Destination A and C 13

Figure 5: Destination A and C 13

The Water: 14

Figure 6: Playing Field “Water area” (Gangplank not shown) 14

Destination B 15

Figure 7: Destination B 16

Green Indicator LED 17

Pirate Flag Destination 17

Treasure Chest 18

Figure 8: Treasure Chest 18
Scoring

Two Qualifying Rounds: Round 1, and Round 2
Elimination round: Round 3

Terminology

**Pirate:** Robot

**Ship:** Raised portion of the field

**Island:** The lower portion of the field not painted blue

**Doubloons:** Points

**Gangplank:** Ramp from the ship to the island

**Treasure Map:** The string of 3 binary bits transmitted by the infrared sensor

**Destination A / Lowering the bridge:** Push button that lowers the bridge

**Destination B / Location of treasure key:** Pressure plate

**Destination C / Raising the bridge:** Push button that raises the bridge

**Treasure Chest:** The conduit box filled with loose weights

**Pirate Flag:** The flag raised by the captain’s wheel and provided with the field

Avast, Me Hearties!

A pirate begins the journey by reading a treasure map and lowering a gangplank onto the island.

The pirate then crosses a narrow gangplank being careful not to fall into the shark infested waters. Once on the island, the map shows the path to the key; then the treasure chest can be claimed! While on the island, the pirate can raise their pirate flag to claim it for the crew. The pirate picks up the treasure chest, returns to the ship, and sets sail on the high seas.

Overview

The major tasks to complete are: reading the treasure map, lowering the gangplank, recovering the key, loading the treasure chest, raising the pirate flag, returning to the ship, and raising the gangplank. The route taken while completing these tasks depends on randomly selected treasure map coordinates.

Pirates have the option of completing all the tasks listed in the coordinates in order, or forgo some tasks in the interest of strategy. The end goal is to collect the maximum number of doubloons in the least amount of time.
The Treasure Map

At the start of a round, the robot receives coordinates for the route that must be taken to the three major destinations. The coordinates are sent as an infrared (IR) signal in an 8 bit message using a protocol based on NEC. The last 3 bits contain the coordinates. At the beginning of each round a code will be randomly generated that teams must be displayed on the robot in order to earn doubloons. Each binary bit represents one of the three destinations (Figure 1). Only destinations listed by the coordinates will earn doubloons. All combinations of destination routes are symmetrical, making all possible combinations equal in length.

Figure 1: SoutheastCon18 Treasure Map

The Route

The robot starts over an infrared LED (Figure 2) where it receives an 8-bit signal. In order to earn Doubloons for properly reading the signal it must be displayed on the robot as a decimal number corresponding to the route number displayed in Table 1. This display may be either a 7-segment display or LCD screen and must be in a clearly visible position. See Table 10 for accepted display examples.
Before the round starts, the IR LED emits a positioning signal to ensure teams are aligned properly to receive the map signal. This alignment signal is sent as logical 1s for all of the 8 bits. This signal will repeat on a loop until the round begins.

The timer starts after a 3, 2, 1, GO countdown from a judge, after which the signal changes from the positioning signal to a randomly generated route signal transmitted from the IR LED. Teams must activate their robots manually when the judge says GO to begin the round. The robot must be placed in the correct starting position and turned on before the transmission occurs. The message will be sent continuously every 200 milliseconds for the first 30 seconds of the round.

There are two possible locations: North (0) and South (1), for each destination (Figure 1). The route taken depends on the coordinates the robot receives (see Table 1). The robot must go to each destination in alphabetical order: A-B-C. For example: if Destination B is activated before completing Destination A, the doubloons from Destination A can no longer be earned, and Destination C is active and ready.

At Destination A, the robot presses a limit switch to “lower the gangplank” and cross the water by using the gangplank to get to the island. Successful completion of Destination A, determined by the green indicator LED turning on, indicates a robot may cross the gangplank for the full point value.

At Destination B, the robot will find the key to unlock the treasure chest by depressing a lever to activate a limit switch. The green indicator LED above the lever (on the 2x4 behind the lever) will confirm successful pad activation.

For maximum doubloons the robot must retrieve the treasure chest and raise the pirate flag. The robot can earn doubloons by moving the treasure chest out of an orange outline that is the exact dimensions of the chest itself. More Doubloons will be awarded for picking up the chest and storing it on the robot (see Table 10 for scoring details). If a robot manages to pick up the chest and finish the round with both the robot and the chest on the ship, additional doubloons are awarded. The treasure chest is located at the same position for every match: inside of the orange outline with the flat, secured steel blank side facing up.

Additionally, the robot may rotate a wheel in the clockwise direction to raise the pirate flag. Rotating the wheel 2.75 times earns a small number of doubloons, while 4.75 times results in the maximum number of doubloons. The RGB LED built into the rotary encoder begins the match illuminated red. Any value between 2.75 to 4.74 turns and the LED illuminates yellow to indicate partial Doubloons. Any value between 4.75 turns and 5.24 turns results in full doubloon value and a green LED. If the wheel is turned 5.25 turns it resets back to zero turns, changing the LED back to red and awarding zero doubloons. Once the LED turns back to red, this effectively resets the turns and needs to turn it 2.75 times for partial doubloons and 4.75 for full doubloons. Doubloons are awarded for the last position left on the flag at the end of the match. The treasure chest and pirate flag are optional stages to earn extra doubloons. However, the pirate flag may be raised at any point during the match.

At Destination C, the robot presses a limit switch to signal the end of the round. Destination C utilizes the same buttons as Destination A. Initially the arcade button is configured as Destination A. The arcade button changes to Destination C 45 seconds after the start of the match or 20 seconds after Destination A(0) or A(1) is pressed. If less than 20 seconds remain of the initial 45 seconds, the remaining time is used instead of 20 seconds. Once Destination B is pressed the arcade button immediately changes to Destination C. The is no deduction of doubloons if the incorrect button is pressed for a Destination.
However, the round ends on the first press of Destination C(0) or C(1) even if the wrong Destination C is activated.

A green LED is located above each major destination (A,B,C). For Destination A/C the indicator LEDs are located at 1.5” and 4.5” from the left on the top edge of the plywood in a 3D printed mount from the CAD files. When viewing the button the Destination A LED is located on the left, and Destination C LED is located on the right. The indicator LED is illuminated if the correct station was activated. If the wrong station was pressed the LED will not illuminate. The LEDs themselves may be used by robots to indicate that the correct button was pressed; there is a diffuser over the LEDs to better distinguish them.

<table>
<thead>
<tr>
<th>Route</th>
<th>Code (Right=LSB)</th>
<th>Destination A Location</th>
<th>Destination B Location</th>
<th>Destination C Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 0</td>
<td>A (0)</td>
<td>B (0)</td>
<td>C(0)</td>
</tr>
<tr>
<td>2</td>
<td>0 0 1</td>
<td>A (0)</td>
<td>B (0)</td>
<td>C(1)</td>
</tr>
<tr>
<td>3</td>
<td>0 1 0</td>
<td>A (0)</td>
<td>B (1)</td>
<td>C(0)</td>
</tr>
<tr>
<td>4</td>
<td>0 1 1</td>
<td>A (0)</td>
<td>B (1)</td>
<td>C(1)</td>
</tr>
<tr>
<td>5</td>
<td>1 0 0</td>
<td>A (1)</td>
<td>B (0)</td>
<td>C(0)</td>
</tr>
<tr>
<td>6</td>
<td>1 0 1</td>
<td>A (1)</td>
<td>B (0)</td>
<td>C(1)</td>
</tr>
<tr>
<td>7</td>
<td>1 1 0</td>
<td>A (1)</td>
<td>B (1)</td>
<td>C(0)</td>
</tr>
<tr>
<td>8</td>
<td>1 1 1</td>
<td>A (1)</td>
<td>B (1)</td>
<td>C(1)</td>
</tr>
</tbody>
</table>
Figure 2: SoutheastCon18 Playing Field

Destination A & C: Button, Destination B: Treasure Key**, Destination C: Button

*The laser hits a LDR mounted on the bridge
**The treasure key is a lever
***The robot needs to turn the wheel to raise the flag
Playing Rules

The objective is to collect doubloons (see Table 10 for doubloons breakdown) by completing objectives. Scores can be influenced by how a team wants to play the game. The more tasks a team completes, the more doubloons they collect (see Table 10 for scoring details).

- Robot must be autonomous - No wired or wireless communications during competition other than IR receiver mounted on the bottom of the robot to receive coordinate location.
- Robot must remain a single unit and can not be modular.
- Robots may be no larger than 12”x12”x12” at the start of the match.
- Robots may not extend outside of 20”x20”x20” during a match. Robots may end a match at these dimensions.
- Each team competes in two rounds on Saturday, April 21st before a final round during the Banquet.
- The scoring for each round is cumulative, with the score from the second round being added to the score of the first round.
- The final round with the top four scoring teams will be held on Saturday, April 21st during the awards banquet.
- The treasure chest will be set in the same position covering the entirety of the orange square in the centerline of the island at the beginning of each round. The chest will be oriented so that the flat steel blank screwed into the chest is facing upwards.
- The captain’s wheel will be oriented so that two of the eight spokes are perpendicular and two are parallel to the surface of the island.
- The treasure chest and captain’s wheel will be set visually.
- Robots must be present at the Starting sequester location beginning from 30 minutes to 15 minutes before the beginning of a round.
- If a team has not reported for sequestering prior to the 15 minute limit, that team will not be allowed to compete in that round, and may only earn doubloons collected outside of round (eg. team shirts, logo and flag).
- Flags will be collected during the sequester time of the first match and returned to their respective teams at the end of the second match, with the exception of the teams competing in the final. The flags of the four teams competing in the final will be returned at the end of the finals. At the beginning of each match the team flag will be placed into a holder on the outside of the field.
- Only 2 team members are allowed within the designated playing area at a time. Violations will result in a 250 doubloon deduction for the round.
- The team members in the playing area (robot handlers) must wear the team shirt to earn shirt Doubloons.
- Team shirts and flags used for hardware competition must display the same team logo to earn doubloons credit.
- No two teams may occupy a playing field at the same time.
- Judges will initiate the start of a round.
- Rounds will begin with a judge counting “3, 2, 1, GO” teams must manually start their robot when the judge says GO.
- When the wrong destination is pressed, the indicator LED will not activate.
- Teams can only receive Doubloons for successfully activating a destination once.
● A laser will be used to detect and credit doubloons when a robot has crossed the gang plank.
● Teams must not drop the treasure chest more than 6 inches from the top of the ¼” plywood field base that the robot’s base is currently on, or intentionally launch the chest.
● Damaging the field will result in a 200 point deduction or disqualification depending on the severity.
● Robots cannot contact the water at any point during the match. Lasers will detect the robot crossing the water edge. A round can also be ended if a judge sees the robot contact the water, but the robot corrects before crossing over the laser beam. Any doubloons earned up to that point will be kept.
● Any teams found to be deliberately exploiting the field’s automatic scoring system, firing projectiles, interfering with the microcontrollers, the IR sensor, etc. will be removed from the competition.
● Team members are allowed to end the round at any time by having a team member make contact with their robot before interacting with Destination C. Doubloons for time will be counted towards the team's’ overall score.
● A round ends when the robot presses the Destination C button (whether it is the correct button or not), the time reaches the four minute limit, the robot crosses into the water, or when a team member makes physical contact with the robot.
● A team making contact during play or tampering with another team’s robot will result in the offending teams’ disqualification and zero doubloons earned for that round.
● At the end of a round teams must place robots in the End sequester location.
● Robots will be released from sequestering within 15 minutes of all teams completing Rounds 1 and 2 to resolve any appeals.
● Teams will have a 15 minute window after their final score is displayed to file an appeal with the judges if they believe a scoring error exists.
● Teams should use caution in filing appeals. Unsubstantiated appeals will cause a 200 point deduction from the Team filing the appeal if, after review, the Judges determine that the original scoring was correct.
● Judges decisions are final.

Violations of IEEE code of ethics and code of conduct will not be tolerated and will result in point deduction, disqualification, or ejection from the event based on the severity of the violation.

When addressing judges with questions teams are expected to act within the IEEE code of conduct.

Doubloon earning locations
● Start Pad (when display shows correct path)
● Time - Each team starts with 240 doubloons and loses one per second until the end of the round
● Button (Destination A) - “lowers gangplank”
● Crossing Bridge
● Key (Destination B)
● Treasure chest
● Pirate flag being raised
● Button (Destination C) - Return to Ship and raise the gangplank
Doubloons earned outside of round

- Team Shirts - To earn doubloons, Hardware Team handlers must wear Team Shirts throughout hardware competition.
- Team Flag - To earn doubloons, teams must provide a flag to be placed on the ship section of the field during a team's run; must include school logo/colors (only if the team is affiliated with a school).
- Team Logo - To earn doubloons, the same logo must be displayed on the team flag, team shirts, and robot.

Robot Specifications

The robot must not be greater than 12”x12”x12” at the beginning of a match. It cannot extend to more than 20”x20”x20” during a match. The robot must not include any form of wireless communication, cannot fly, and cannot make use of projectiles.

Team Logo Specifications

In order to earn full credit for the team logo it must be displayed on the team flag, team shirts, and visible on the robot. If the team is from a school the team logo must incorporate a version of their school logo/mascot. Otherwise if not affiliated with a school no creative design specifications are required. Any imagery that violates the IEEE code of ethics or conduct is not permitted.

Team Flag Specifications

The team flag must display the team logo, must be no larger than 8”x12” (excluding flagpole), and may be any shape within those dimensions. The team flag must be constructed from laminated paper, fabric, or cardstock. Flags constructed of other material such as: un laminated paper, tissue cloth, etc. will not be accepted for Doubloons credit. Flags must be attached to a flagpole no longer than 2 feet and no wider than 1 inch in diameter. The team flag is separate from the flag raised by the captain’s wheel stage, which is provided.

Field Specifications

**NOTE:** All numerical specs for the field are listed or can be calculated from the provided CAD file at ieeesoutheastcon2018.slack.com. Email victorialepp@mail.usf.edu for an invitation to the Slack forum for the file and group Q&A.

The field will be 8’x4’ area.

1 count: Plywood Sheet 4’ x 8’ ¾ thickness
1 count: Plywood Sheet 4’ x 4’ ¾ thickness
4 count: 8’ 2x4
2” and 3” Decking screws

3D printed components will be constructed using PLA with 15% infill, all components except 2 LED holders with be colored Black. The 2 LED holders mounted over Destination B will be colored White. **Paint Specifications (on D, treasure chest is covering the orange paint)**

<table>
<thead>
<tr>
<th>Colour</th>
<th>Specific Name</th>
<th>ID Number</th>
<th>Brand</th>
<th>Location, Figure 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Hills of Ireland</td>
<td>M390-7^D</td>
<td>BEHR</td>
<td>A</td>
</tr>
<tr>
<td>Brown</td>
<td>Burnt Toffee</td>
<td>S230-6^D</td>
<td>BEHR</td>
<td>B</td>
</tr>
<tr>
<td>Blue</td>
<td>Azure Afternoon</td>
<td>89BG 37/353</td>
<td>Glidden</td>
<td>C</td>
</tr>
<tr>
<td>Orange</td>
<td>Jack O Lantern</td>
<td>89YR 36/694</td>
<td>Glidden</td>
<td>D</td>
</tr>
<tr>
<td>Yellow</td>
<td>English Daisy</td>
<td>P290-6^D</td>
<td>BEHR</td>
<td>E</td>
</tr>
<tr>
<td>Black</td>
<td>Black^D</td>
<td>TC-45</td>
<td>BEHR</td>
<td>F</td>
</tr>
<tr>
<td>White</td>
<td>Gloss White</td>
<td>39DP14</td>
<td>Rust-Oleum</td>
<td>G</td>
</tr>
</tbody>
</table>

Figure 3: Playing Field Paint Colors

Initial Signal

Coordinates are sent as an IR signal with a carrier frequency of 38kHz in the following format: The message starts with an initial burst of 9ms, a space of 4.5ms, 8 bits that contain the message, and an ending burst. The first 5 bits of the message will be sent as logical 0s, and the last 3 bits will define the coordinates to follow based on logical 1s and 0s as detailed in the possible routes listed in table 1. The
coordinates start from the left-most bit with A as the first one, followed by B and C. The message will be sent continuously every 200 milliseconds for the first 30 seconds of the round.

The positioning signal will send logical 1s for all 8 bits and should not be recognized as a map signal by robots. The purpose of this signal is to allow teams to align their robots to the output of the IR LEDs and ensure the robot will be able to properly receive the coordinate signal.

**NOTE:** The format is similar to the NEC protocol, however this was just taken as a reference to determine the duration of the pulses for the logical 1s and 0s.
Table 3: Initial Signal Parts

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Part Number/Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED - Infrared 950nm (Initial Signal) (x4)</td>
<td>Model#: COM-09349 ROHS</td>
</tr>
<tr>
<td></td>
<td><a href="https://www.sparkfun.com/products/9349">https://www.sparkfun.com/products/9349</a></td>
</tr>
</tbody>
</table>

Destination A and C

Destination A and C utilize the same button. This button is a round, arcade style button on a 6” x 12” piece of ¾” plywood. This piece of plywood is centered 12” from the outside Western edge of the field with 9” above the top of the ship. The button is centered 6” above the top of the ship and on the
centerline of the short side of the 6” x 12” plywood piece. The green indicator LEDs are located at 1.5” and 4.5” on the top edge of the plywood piece. The indicator LEDs for Destination A will always be on the left when viewing the button.

Figure 5: Destination A and C
<table>
<thead>
<tr>
<th>Part Description</th>
<th>Part Number/Details</th>
</tr>
</thead>
</table>
| Destination A/B Button | Model#: 1568-1476-ND  

The Water:

The area of water is located on the main playing board, east of the ship. The water zone is displayed in Figure 6. Lasers will be positioned as seen in Figure 2 to detect when a robot has crossed into the water.

Figure 6: Playing Field “Water area” (Gangplank not shown)
Table 5: The Water Materials

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Part Number/Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light Detection Signal Switch (x4)</td>
<td>Wangdd22 - LYSB01E6W0HPU-ELECTRNCS <a href="https://www.amazon.com/gp/product/B01E6W0HPU/ref=oh_aui_detailpage_o01_s00?ie=UTF8&amp;psc=1">https://www.amazon.com/gp/product/B01E6W0HPU/ref=oh_aui_detailpage_o01_s00?ie=UTF8&amp;psc=1</a></td>
</tr>
<tr>
<td>Mini Lasers (x4)</td>
<td>Ketofa WYHP <a href="https://www.amazon.com/gp/product/B00R73MC1S/ref=oh_aui_detailpage_o01_s00?ie=UTF8&amp;psc=1">https://www.amazon.com/gp/product/B00R73MC1S/ref=oh_aui_detailpage_o01_s00?ie=UTF8&amp;psc=1</a></td>
</tr>
</tbody>
</table>

Destination B

Destination B is a 6” x 2.25” rectangle made from ¾” plywood that rests on top of a limit switch. The hinge is centered on the 6” x 2.25” plywood piece on the long side as shown in Figure 7. The ¾” plywood+hinge assembly is centered 42” from the East outermost part of the field. From the inside edge of the 2x4, the plywood portion of the assembly is 4” away, and the hinge portion of the assembly is 3” away (see Figure 7).

The limit switch is mounted on the 2” side of a 2” x 1.625” piece of ¾” plywood; this piece is centered at 42” from the East outermost part of the field flush against the 2x4 so that the hinged ¾” plywood rests on the switch. The switch will activate when approximately 0.5 lb of weight is applied to the top edge of the 6” x 2.25” plywood piece. The attached screws shown in Figure 7 are used to protect the switch from any sudden forces applied to it. The screws are mounted 0.5” from the corners of the plywood and raised approximately 0.3” from each surface. It is recommended that the screws be raised to just barely allow the button to be fully activated but not fully compressed.

Table 6: Destination B Parts

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Part Number/Details</th>
</tr>
</thead>
</table>
Green Indicator LED

A green-colored LED will be located above each objective of Destination A, B, and C. The LED will light up only if the correct objective is pressed based on the initial signal received at the start of the round. If the wrong objective is activated, the LED will not illuminate, signifying that the wrong objective was pressed. The indicator LED for Destination B is mounted on the top of the northern and southern 2x4s centered at 42”. The indicator LEDs holders for Destination A/C will be printed using Black colored PLA, and Destination B will be printed using White PLA.

Figure 7: Destination B

Note: Measurements are correct, but scale is not.
### Table 7: Indicator LED Parts

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Part Number/Details</th>
</tr>
</thead>
</table>

### Pirate Flag Destination

The pirate flag will be raised via a 3D printed pirate ship wheel attached to a rotary encoder. The 3D printed wheel will slip over the clear knob supplied with the rotary encoder; the light from the LED will remain visible through the clear knob. The encoder will be located on a 10” x 15” piece of ¾” plywood. The rotary encoder is located 10.25” from the plywood surface when mounted on the edge of the field. This piece is centered 24” from the Northern edge of the field, mounted on the outside of the 2x4 with 10.875” of the board above the Eastern 2x4. The rotary encoder is centered 7” from the top of the 2x4 and on the centerline of the short side of the 10” x 15” plywood piece.

### Table 8: Pirate Flag Destination Parts

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Part Number/Details</th>
</tr>
</thead>
</table>

### Treasure Chest

The treasure chest is a 4” x 4” x 2½” electrical conduit box filled with eight, 4 ounce pyramid fishing sinkers. These weights are loose inside the box. The chest is sealed with a 4” square blank cover. For each round the chest is located 18” from the edge of the chest to the inside of the eastern wall, and 20.5” from the Northern and Southern inside edges.
Table 9: The Treasure Chest Materials

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Part Number/Details</th>
</tr>
</thead>
</table>
| 4 in. 30.3 cu. in. Steel Square Electrical Box | Model#: 521711-25R  
[http://www.gordonelectricsupply.com/index~text~599644~path~product~part~5999644~ds~dept~process~search?gclid=CjwKCAjwyIHPBRAIEiwAHP S-GMFsh-uDU/Nxt3RlMUrYkbi1qOmvYKBM3uRzLgS76E7sAl9_yp1vapRoC-YMQAtvBwE](http://www.gordonelectricsupply.com/index~text~599644~path~product~part~5999644~ds~dept~process~search?gclid=CjwKCAjwyIHPBRAIEiwAHP S-GMFsh-uDU/Nxt3RlMUrYkbi1qOmvYKBM3uRzLgS76E7sAl9_yp1vapRoC-YMQAtvBwE) |
| 4 in. Square Blank Cover, Flat         | Model#: 8752  
| 8x 4oz. Pyramid sinker weights         | Model#: PYR-4  
[https://www.amazon.com/South-Bend-PYR-4-Pyramid-Sinker/dp/B003QAOKIQ/ref=sr_1_1?ie=UTF8&qid=1507318744&sr=8-1&keywords=4+ounce+pyramid+sinker](https://www.amazon.com/South-Bend-PYR-4-Pyramid-Sinker/dp/B003QAOKIQ/ref=sr_1_1?ie=UTF8&qid=1507318744&sr=8-1&keywords=4+ounce+pyramid+sinker) |

Scoring

Teams earn doubloons based on completing tasks and actions as specified in the Playing rules, Table 8, and Table 10.
Two Qualifying Rounds: Round 1, and Round 2

1. All qualifying rounds are scored by the same criteria.
2. There are two identical competition fields and four practice fields. Teams will compete at the same time on separate fields.
3. Rounds are 4 minutes long; rounds start when a judge counts down “3, 2, 1, GO” and initiates the field to send the first IR signal containing the route. A round ends when any Destination C button is pressed, when a team member contacts the robot, the timer runs out, or the robot crosses into the water.
4. Each team will have a randomly generated set of coordinates sent to their robots via the IR LED each round.

Elimination round: Round 3

1. The top 4 teams with the highest combined score from the qualifying rounds will proceed to the final round.
2. The top 4 teams will start at a base score of zero entering the final round. No Doubloons earned from the qualifying rounds will be counted in the final round. For the teams who do not qualify, their final standings will only be the combined scores from the qualifying rounds.
3. The final standings for the top 4 teams will show their score from the final round in parentheses, followed by the combined score from the qualifying round. Example final scores (final round score if applicable, combined qualifying score):
   1) (1200) 1400
   2) (1100) 1400
   3) (900) 1400
   4) (500) 1500
   5) 1300
   6) 1200
   7) 1200
4. The team with the highest score from the final round is the competition winner.
5. Finals will occur during the banquet on Saturday, April 21st.
<table>
<thead>
<tr>
<th>Action</th>
<th>Doubloons</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time remaining</td>
<td>240 to 0</td>
<td>Doubloons will be issued based on the number of seconds the team has remaining after completing the round.</td>
</tr>
<tr>
<td>Formula:</td>
<td></td>
<td>Doubloons earned = 240 - Completion time (seconds).</td>
</tr>
<tr>
<td>Display Correct Code</td>
<td>200</td>
<td>Correct decimal code displayed on LCD or 7-segment display. A period must be displayed after the number. Teams are displaying the <strong>route number</strong> (see Table 1), not the binary number.</td>
</tr>
<tr>
<td>Examples:</td>
<td></td>
<td>Received Binary: 000 Displayed Decimal: 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Received Binary: 011 Displayed Decimal: 4.</td>
</tr>
<tr>
<td>Activate the Correct Destination A</td>
<td>50</td>
<td>Press the correct button on Destination A</td>
</tr>
<tr>
<td>Cross gangplank from ship</td>
<td>100</td>
<td>Crossing the gangplank and the indicator LED from Destination A is green (the correct button was pressed)</td>
</tr>
<tr>
<td>Cross gangplank from ship without active Destination A</td>
<td>50</td>
<td>Crossing the gangplank while the indicator LED is not on.</td>
</tr>
<tr>
<td>Activate Destination B</td>
<td>200</td>
<td>Maneuver robot onto the correct lever.</td>
</tr>
<tr>
<td>Action Description</td>
<td>Points</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Move Treasure Chest</td>
<td>100</td>
<td>Move the treasure chest so that at least ¼” of the orange square under the chest is visible. The ¼” will be measured from the edge of the orange square to the furthest moved point of the chest. If the chest is picked up after moving it, only the 200 Doubloons for picking up the chest will be awarded.</td>
</tr>
<tr>
<td>Pick up and store Treasure Chest in/on robot</td>
<td>200</td>
<td>Successfully picking up and storing the treasure chest in or on the robot. The chest must be lifted so that no part is in contact with the ¾” plywood surface, high enough for a judge to determine it is no longer in contact with the field, and must remain lifted for at least 5 seconds. If the chest is dropped after the 5 second period the 200 Doubloons is still honored.</td>
</tr>
<tr>
<td>Finish on ship with treasure chest</td>
<td>300</td>
<td>Finish on the ship with the treasure chest and robot completely within the confines of the ship.</td>
</tr>
<tr>
<td>Fully Raised Flag Position</td>
<td>200</td>
<td>Raise the flag to the top of the flagpole by turning the captain's wheel clockwise. Teams may turn the wheel as many times as they like. The final flag position and lit LED color as the robot releases contact will be awarded the corresponding Doubloons. 4.75 to 5.24 turns: 200 Doubloons 5.25 =&gt; 0 to 2.74 turns: 0 Doubloons</td>
</tr>
<tr>
<td>Partially Raised Flag Position</td>
<td>100</td>
<td>Raise the flag to halfway up flagpole by turning the captain's wheel clockwise. Teams may turn the wheel as many times as they like. The final flag position and the lit LED color as the robot releases contact will be awarded the corresponding Doubloons. 2.75 to 4.74 turns: 100 Doubloons 0 to 2.74 turns: 0 Doubloons</td>
</tr>
<tr>
<td><strong>Activate the Correct Destination C</strong></td>
<td><strong>100</strong></td>
<td>Finish in the correct location by pressing the correct button.</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Team Shirts</strong></td>
<td><strong>50</strong></td>
<td>To earn doubloons, Hardware Team handlers must wear Team Shirts throughout hardware competition. 50 doubloons will be issued for every round.</td>
</tr>
<tr>
<td><strong>Team Flag</strong></td>
<td><strong>50</strong></td>
<td>To earn doubloons, team must provide a Team Flag with their team logo. Teams should have at least one flag which will be turned in at robot check in.</td>
</tr>
<tr>
<td><strong>Team Logo</strong></td>
<td><strong>50</strong></td>
<td>To earn doubloons, must match Team Flag and be visibly identifiable on the robot. 50 doubloons will be issued for each round played.</td>
</tr>
</tbody>
</table>
Appendix B: Source Code

```c
#include "stm32f407 vg.h"
#include <stdio.h>
#include <stdlib.h>

//global variables
int encoder_fr;
int encoder_br;
int encoder_fl;
int encoder_bl;
int encoder_Flag;
int encoder_avg = 0;
unsigned char newData = 0;
unsigned char readMsg[5];
int length;
int dist;
int flag = 0;
int four_mil = 0;
int dist1, dist2;
int edge;
int one_mil;
int time;
int route;
char tmp[50];
int lower = 8;

//functions
void startup (void);
void move_right(int, char);
void move_left(int, char);
void move_forward(int, char);
void move_back(int, char);
void move_cw(void);
void move_ccw(void);
void flag_cw(void);
void print(char* msg);
void printLCD(char* msg);
int ping_RF(void);
int ping_RB(void);
int ping_LF(void);
int ping_LB(void);
int ping_FR(void);
int ping_FL(void);
int ping_BR(void);
int ping_BL(void);
void clear_encoders(void);
void level_right(void);
void level_left(void);
void level_front(void);
void level_back(void);
void speed (int);
void center(void);
void pickup(void);
```
void TA_out(void);
void TA_in(void);
void LA_left(void);
void LA_right(void);
int read_IR(void);
void destA0(void);
void destA1(void);
void destB0(void);
void destB1(void);
void flag_chest(void);
void destC0(void);
void destC1(void);
void turn_right(void);
void turn_left(void);
void forward_enc(int, int);
void back_enc(int, int);
void right_enc(int, int);
void left_enc(int, int);
void flag_ccw(int);

int main()
{
startup();

route = read_IR() + 1;

if(route==1)
{
    sprintf(tmp, "%s Route %d. (000)", route);
    printLCD(tmp);
    destA0();
    destB0();
    flag_chest();
    destC0();
}
else if(route==2)
{
    sprintf(tmp, "%s Route %d. (001)", route);
    printLCD(tmp);
    destA0();
    destB1();
}
else if(route==3)
{
    sprintf(tmp, "%s Route %d. (010)", route);
    printLCD(tmp);
    destA0();
    destC1();
}
else if(route==4)
{
    sprintf(tmp, "%s Route %d. (011)", route);
    printLCD(tmp);
    destA0();
    destB0();
}
flag_chest();
destC0();
sprintf(tmp, "/* Route %d. (010)", route);
printLCD(tmp);
}

else if(route==4)
{
    sprintf(tmp, "/* Route %d. (011)", route);
    printLCD(tmp);
destA0();
destB1();
flag_chest();
destC1();
sprintf(tmp, "/* Route %d. (011)", route);
printLCD(tmp);
}

else if(route==5)
{
    sprintf(tmp, "/* Route %d. (100)", route);
    printLCD(tmp);
destA1();
destB0();
flag_chest();
destC0();
sprintf(tmp, "/* Route %d. (100)", route);
printLCD(tmp);
}

else if(route==6)
{
    sprintf(tmp, "/* Route %d. (101)", route);
    printLCD(tmp);
destA1();
destB0();
flag_chest();
destC1();
sprintf(tmp, "/* Route %d. (101)", route);
printLCD(tmp);
}

else if(route==7)
{
    sprintf(tmp, "/* Route %d. (110)", route);
    printLCD(tmp);
destA1();
destB1();
flag_chest();
destC0();
sprintf(tmp, "/* Route %d. (110)", route);
printLCD(tmp);
}

else if(route==8)
{
    sprintf(tmp, "/* Route %d. (111)", route);
    printLCD(tmp);
destA1();
destB1();
flag_chest();
destC1();
sprintf(tmp, "/* Route %d. (111)", route);
printLCD(tmp);
}
void startup (void) {

  // Enc Wheels and Flag A0-5 (A0-BL, A1-FL, A2-BR, A3-FR, A4-FLAG1, A5-FLAG2) 177  //IR Signal A6
  // Button Linear Actuators A7-8 (A7-DestA0/left, A8-DestA1/right)
  // ADC Track Actuator A9
  // Dir C0-5 (C0-BR, C1-FR, C2-BL, C3-FL, C4-FLAG, C5-TRACK) (0-backward/in, 1-forward/out) (1-CW, 0-CCW) 181  //Dis MBED C6-7 (C6-Tx, C7-Rx)
  // LCD Screen B6
  // PWM Servo Chest and Flag (B5-SERVO (CH2), B0-LAFLAG (CH3), B1-MotorFlag (CH4))
  // Ports not allowed to be used: B13-15, C8

  // Enable clocks
  RCC_AHB1ENR |= 0xF;  // Enable PA-D clocks
  RCC_APB1ENR |= (1 << 2);  // Peripheral timer 4 clocks
  RCC_APB1ENR |= (1 << 1);  // Peripheral timer 3 clocks
  RCC_APB1ENR |= (1 << 0);  // Peripheral timer 2 clocks
  RCC_APB2ENR |= 0x100;  // Bit 8 is ADC 1 clock enable bit

  // Port A setup
  GPIOA_MODER |= 0xD4000;  // Set bits PC0-5 to output
  GPIOA_OTYPER |= 0x000;  // Set bits 0-5 to output push-pull (reset state)
  GPIOA_OSPEEDER |= 0xAA;  // Set bits 0-5 to high speed

  // Port B setup (PWM)
  GPIOB_MODER &= 0x00F0;  // Clear PB0-1 and PB4-5
  GPIOB_MODER |= 0xA0A;  // Set bits 0-1 and 4-5 to alternate function
  GPIOB_OTYPER &= 0xCFC;  // Set bit 0-1 and 4-5 to push/pull
  GPIOB_OSPEEDER &= 0x0F0;  // Set bit 0-1 and 4-5 to low speed
  GPIOB_AFRL &= 0x220022;  // Clear alternate function register low
  GPIOB_AFRL |= 0x220022;  // Set bits 0-1 and 4-5 to alternate function two

  // Port C setup
  GPIOC_MODER |= 0x555;  // Set bits PC0-5 to output
  GPIOC_OTYPER |= 0x000;  // Set bits 0-5 to output push-pull (reset state)
  GPIOC_OSPEEDER |= 0xA0;  // Set bits 0-5 to high speed

  // Port D setup (PWM)
  GPIOD_MODER &= 0x00000000;  // Clear PD12-15
  GPIOD_MODER |= 0xAA000000;  // Set bits 12-15 to alternate function
  GPIOD_OTYPER &= 0x0FF;  // Set bit 12-15 to push/pull
  GPIOD_OSPEEDER &= 0x0F000000;  // Set bit 12-15 to low speed
  GPIOD_AFRH &= 0x22220000;  // Clear alternate function register high
  GPIOD_AFRH |= 0x22220000;  // Set bits 12-15 to alternate function two
//External Interrupts
RCC_APB2ENR |= 0x4000;  //SYSCFG clock enable
SYSCFG_EXTICR1 &= 0xFFFF0000;  //EXTI0-3 set to PA0-3
SYSCFG_EXTICR2 &= 0xFFFF0000;  //EXTI4-7 set to PA4-7
NVICISER0 |= 0x7C0;  //Enable EXTI0-4
NVICISER0 |= (1<<23);  //Enable EXTI5-9
EXTI_IMR |= 0x5F;  //Don't mask interrupt line 0-4 and 6
EXTI_FTSR |= 0x1F;  //Falling trigger enable line 0-5
EXTI_RTSR |= 0x40;  //Rising trigger enable line 6-7

//Set up timer 2
NVICISER0 |= (1 << 28);  //Enable timer 2
TIM2_DIER |= 1;  //Update interrupt enable
TIM2_DIER |= (1 << 6);  //Trigger enable interrupt
TIM2_CR1 |= (1 << 7);  //Auto reload enable
TIM2_PSC = 0;  //Timer prescaler
TIM2.ARR = 4000;  //Auto reload register
TIM2_EGR |= 1;
TIM2.CR1 |= 1;  //Timer enabled

//Set up timer 3
NVICISER0 |= (1 << 29);  //Enable timer 3
TIM3_DIER |= 1;  //Update interrupt enable
TIM3_DIER |= (1 << 6);  //Trigger enable interrupt
TIM3_CR1 |= (1 << 7);  //Auto reload enable
TIM3_PSC = 0;  //Timer prescaler
TIM3.ARR = 64000;  //Auto reload register
TIM3.CCMR1 |= 0x6868;
TIM3.CCMR2 |= 0x6868;  //PD14 and PD15 are channels 3 and 4, Selects PWM mode 1 for both capture 3 and 4 modes and enables compare 3 and 4 modes
TIM3.CCER |= 0x1111;  //Capture/Compare 3 and 4 output enable
TIM3.CCR2 = 0;  //PB5(SERVO) duty cycle (16000-24320-32000)
TIM3.CCR3 = 0;  //PB0(LAFLAG) duty cycle
TIM3.CCR4 = 0;  //PB1(MotorFlag) duty cycle
TIM3.EGR |= 1;
TIM3.CR1 |= 1;  //Timer enabled

//Set up timer 4
NVICISER0 |= (1 << 30);  //Enable timer 4
TIM4_DIER |= 1;  //Update interrupt enable
TIM4_DIER |= (1 << 6);  //Trigger enable interrupt
TIM4.CR1 |= (1 << 7);  //Auto reload enable
TIM4_PSC = 0;  //Timer prescaler
TIM4.ARR = 64000;  //Auto reload register
TIM4.CCMR1 |= 0x6868;
TIM4.CCMR2 |= 0x6868;  //PD14 and PD15 are channels 3 and 4, Selects PWM mode 1 for both capture 3 and 4 modes and enables compare 3 and 4 modes
TIM4.CCER |= 0x1111;  //Capture/Compare 3 and 4 output enable
TIM4.CCR1 = 0;  //PD12(FR) duty cycle
TIM4_CCR2 = 0; //PD13(BR) duty cycle
TIM4_CCR3 = 0; //PD14(FL) duty cycle
TIM4_CCR4 = 0; //PD15(BL) duty cycle
TIM4_EGR |= 1;
TIM4_CR1 |= 1; //Timer enabled

// Clock bits UART6
RCC_AHB1ENR |= 4; //Bit 3 is GPIOC clock enable bit
RCC_APB1ENR |= (1 << 4); //Enable peripheral timer for timer 6
RCC_APB2ENR |= (1 << 5); //Enable USART6 clock
GPIOC_AFRL |= 0x88000000; //Alternate Func PC 6-7 to USART6
GPIOC_MODER |= 0x0A000; //Bits 15-12 = 1010 for Alt Func on PC6, PC7
GPIOC_OSPEEDER |= 0x3000; //Bits 7-6 = 11 for high speed on PC6
GPIOB_AFRL |= 0x77000000; //Alternate Func PB 6-7 to USART1
GPIOB_MODER |= 0x0A000; //Bits 15-12 = 1010 for Alt Func on PB6, PB7
GPIOB_OSPEEDER |= 0x3000; //Bits 7-6 = 11 for high speed on PB6

// Configure UART6
USART6_CR1 = 0; //Disable during set up. Wd len = 8, Parity = off
USART6_BRR = 0x3E8; //Set up baud rate (3E8 for MBED)
USART6_CR2 = 0; //1 stop bit
USART6_CR1 = 0x202C; //Enable transmit and receive and USART and receive interrupt
USART6_CR3 = 0; //Disable interrupts and DMA
NVICISER2 |= (1 << 7); //Enable USART6 interrupt

// Clock bits UART1
RCC_AHB1ENR |= 2; //Bit 1 is GPIOB clock enable bit
RCC_APB2ENR |= (1 << 0); //Enable peripheral timer for timer 1
RCC_APB2ENR |= (1 << 4); //Enable USART1 clock
GPIOB_AFRL |= 0x77000000; //Alternate Func PB 6-7 to USART1
GPIOB_MODER |= 0x0A000; //Bits 15-12 = 1010 for Alt Func on PB6, PB7
GPIOB_OSPEEDER |= 0x3000; //Bits 7-6 = 11 for high speed on PB6

// Configure UART1
USART1_CR1 = 0; //Disable during set up. Wd len = 8, Parity = off
USART1_BRR = 0xD05; //Set up baud rate (D05 for LCD)
USART1_CR2 = 0; //1 stop bit
USART1_CR1 = 0x200C; //Enable transmit and receive and USART and receive interrupt
USART1_CR3 = 0; //Disable interrupts and DMA
NVICISER1 |= (1 << 5); //Enable USART1 interrupt
}
int read_IR()
{
    int route=0;
    float avg;
    while(1)
    {
        while(time<52 || time>60)
        {
            edge=0;
            one_mil=0;
            while(edge<1)
            {
                time = one_mil;
            }
            edge=0;
            one_mil=0;
            while(edge<1)
            {
                time=one_mil;
            }
            if (time<6)
                break;
            edge=0;
            while(edge<7);
            edge=0;
            one_mil=0;
            for(int i=0;i<3;i++)
            {
                one_mil=0;
                while(edge<1)
                {
                    time=one_mil;
                }
                if (time>6)
                    route |= 1 << (2-i);
            }
        }
    } // while(edge<6)
    // time1=one_mil;
    // while(edge<7)
    // time2=one_mil;
    // while(edge<8)
    // time3=one_mil;
    // if (time1>6)
    //    route |= 1 << 2;
    // if (((time2-time1)>6)
    //    route |= 1 << 1;
    // if (((time3-time2)>6)
    //    route |= 1 << 0;

return route;
}

void destA0()
{
    turn_left();
    forward_enc(6000, 80);
    turn_right();
    level_back();
    move_forward(110, 'B');
    move_left(67, 'F');
    move_forward(137, 'B');
    LA_left();
    right_enc(1000, 80);
    back_enc(1000, 80);
    level_back();
    turn_right();
    forward_enc(5000, 80);
    turn_left();
    level_back();
    move_forward(150, 'B');
    forward_enc(15100, 50);
    GPIOA_ODR &= 0x7F;
}

void destA1()
{
    turn_right();
    forward_enc(6000, 80);
    turn_left();
    level_back();
    move_forward(110, 'B');
    move_right(67, 'F');
    move_forward(155, 'B');
    LA_right();
    left_enc(1000, 80);
    back_enc(1000, 80);
    level_back();
    turn_left();
    forward_enc(5000, 80);
    turn_right();
    level_back();
    move_forward(150, 'B');
    forward_enc(15000, 50);
    GPIOA_ODR &= 0xEFF;
}

void destB0()
{
    turn_left();
    forward_enc(7300, 60);
    back_enc(15000, 80);
    turn_right();
    GPIOC_ODR = 0xC;  
    clear_encoders();
    speed(75);  
    while(encoder_avg<500)
    encoder_avg = (encoder_bl+encoder_br+encoder_fl+encoder_fr)/4;
    speed(0);
    forward_enc(10000, 80);
level_front();
move_forward(100, 'R');
turn_right();
dist1 = ping_FR();
if (dist1 < 500)
{
move_forward(100, 'R');
turn_right();
}
forward_enc(4000, 80);
turn_left();
}

void destB1()
{

}

forward_enc(6300, 80);
back_enc(1600, 80);
turn_left();

GPIOC_ODR = 0x3;
clear_encoders();
speed(75);
while (encoder_avg < 300)
encoder_avg = (encoder_bl+encoder_br+encoder_fl+encoder_fr)/4;
speed(0);
forward_enc(10000, 80);
level_front();
move_forward(100, 'L');
turn_left();
dist1 = ping_FR();
if (dist1 < 500)
{
move_forward(100, 'R');
turn_left();
}
forward_enc(5000, 80);
turn_right();
}

void flag_chest()
{
forward_enc(1500, 80);
dist1 = ping_RB();
dist2 = ping_LB();
while (1)
{
four_mil = 0;
if (dist2 > 100)
{
back_enc(300, 15);
right_enc(10, 15); 486
dist1 = ping_RB();
dist2 = ping_LB();
forward_enc(350, 15);
}
else if (dist1 > 100)
{
back_enc(300, 15); 493
left_enc(10, 15);
dist1 = ping_RB();
dist2 = ping_LB();
forward_enc(500, 15);
        }  
        else
        break;
    
    if(four_mil>5000)
    {
        turn_right();
        dist1 = ping_FR();
        if(dist1<400)
        {
            turn_left();
            level_front();
            turn_left();
            forward_enc(3000, 80);
            turn_right();
        }
        else
        {
            forward_enc(3000, 80);
            turn_left();
            four_mil=0;
        }
    }
    
    forward_enc(200, 80);
    TA_out();
    flag_cw();
    right_enc(1000, 50);
    forward_enc(300, 80);
    four_mil = 0;
    while(four_mil<125);
    back_enc(2100, 80);
    
    back_enc(10000, 80);
    move_back(200,'L');
    move_back(50,'L');
  
  
  void destC0()
  {
    move_forward(110, 'B');
    level_back();
    turn_right();
    back_enc(7000, 70);
    back_enc(3000, 40);
    // turn_left();
    // forward_enc(5700, 80);
    // turn_right();
    // move_forward(110, 'B');
    // dist1 = ping_LF();
    // while(1)
    // if(dist1>125)
    //     {
    //         turn_left();
    //         forward_enc(1000, 80);
    //         four_mil = 0;
    //         while(four_mil<125);
    //         turn_right();
    //         level_back();
    //         move_forward(110, 'B');
558 // move_back(110, 'R');
559 // dist1 = ping_LF();
560 // }
561 // else
562 // break;
563 // }
564 // level_back();
565 // move_left(67, 'F');
566 // move_forward(137, 'B');
567 // LA_left();
568 // }
569
570 void destC1()
571 { 
572 move_forward(110, 'B');
573 level_back();
574 turn_left();
575 back_enc(7000, 70);
576 back_enc(3000, 40);
577 // turn_right();
578 // forward_enc(5700, 80);
579 // turn_left();
580 // move_forward(110, 'B');
581 // dist1 = ping_RF();
582 // while(1)
583 // {
584 // if(dist1>125)
585 // {
586 // turn_right();
587 // forward_enc(1000, 80);
588 // four_mil = 0;
589 // while(four_mil<125);
590 // turn_left();
591 // level_back();
592 // move_forward(110, 'B');
593 // move_back(110, 'R');
594 // dist1 = ping_RF();
595 // }
596 // else
597 // break;
598 // }
599 // level_back();
600 // move_right(67, 'F');
601 // move_forward(137, 'B');
602 // LA_right();
603 // }
604
605
606 void clear_encoders()
607 { 
608 encoder_avg = 0;
609 encoder_fr = 0;
610 encoder_br = 0;
611 encoder_fl = 0;
612 encoder_bl = 0;
613 }
void flag_ccw(int ticks) {
    encoder_flag = 0;
    GPIOC_ODR = 0x10;
    TIM3_CCR4 = 32000;
    TIM3_EGR |= 1;
    while (encoder_flag < ticks);
    TIM3_CCR4 = 0;
    TIM3_EGR |= 1;
}

void flag_cw() {
    int ticks;
    encoder_flag = 0;
    GPIOC_ODR = 0x10;
    TIM3_CCR4 = 32000;
    TIM3_EGR |= 1;
    four_mil = 0;
    while (encoder_flag < 20900) {
        if (four_mil > 250 && encoder_flag < 2048) {
            TIM3_CCR4 = 0;
            TIM3_EGR |= 1;
            GPIOC_ODR = 0x0;
            GPIOC_ODR = 0x00;
            ticks = encoder_flag;
            TIM3_CCR3 = 64000;
            TIM3_EGR |= 1;
            four_mil = 0;
            while (four_mil < 350);
            TIM3_CCR3 = 0;
            TIM3_EGR |= 1;
            flag_ccw(ticks);
            while (1) {
                if (dist2 > 100) {
                    back_enc(300, 15);
                    right_enc(10, 15);
                    dist2 = ping_RB();
                    forward_enc(350, 15);
                } else if (dist1 > 100) {
                    back_enc(300, 15);
                    left_enc(10, 15);
                    dist1 = ping_RB();
                    dist2 = ping_LB();
                    forward_enc(500, 15);
                } else {
                    break;
                }
            }
        } else if (dist1 > 100) {
            back_enc(300, 15);
            left_enc(10, 15);
            dist1 = ping_RB();
            dist2 = ping_LB();
            forward_enc(500, 15);
        } else {
            break;
        }
    }
    GPIOC_ODR = 0x20;
    TIM3_CCR3 = 64000;
    TIM3_EGR |= 1;
    four_mil = 0;
    while (four_mil < 350);
TIM3_CCR3 = 0;
TIM3_EGR |= 1;
encoder_flag=0;
four_mil=0;
GPIOC_ODR = 0x10;
TIM3_CCR4 = 32000;
TIM3_EGR |= 1;
}
TIM3_CCR4 = 0;
TIM3_EGR |= 1;
}

void TA_out()
{
    GPIOC_ODR = 0x20;
    TIM3_CCR3 = 64000;
    TIM3_EGR |= 1;
    four_mil = 0;
    while(four_mil<800);
    TIM3_CCR3 = 0;
    TIM3_EGR |= 1;
}

void TA_in()
{
    GPIOC_ODR = 0x00;
    TIM3_CCR3 = 64000;
    TIM3_EGR |= 1;
    four_mil = 0;
    while(four_mil<800);
    TIM3_CCR3 = 0;
    TIM3_EGR |= 1;
}

void LA_left()
{
    four_mil=0;
    GPIOA_ODR |= 0x80;
    while (four_mil<1500);
    GPIOA_ODR &= 0x7F;
    GPIOA_ODR &= 0x7F;
}

void LA_right()
{
    four_mil=0;
    GPIOA_ODR |= 0x100;
    while (four_mil<1500);
    GPIOA_ODR &= 0xEFF;
    GPIOA_ODR &= 0xEFF;
}
void pickup()
{
    for(int rotate=20000;rotate<35000;rotate+=150)
    {
        four_mil=0;
        TIM3_CCR2 = rotate;
        TIM3_EGR |= 1;
        while (four_mil<10);
    }
    for(int rotate=35000;rotate>18000;rotate-=150)
    {
        four_mil=0;
        TIM3_CCR2 = rotate;
        TIM3_EGR |= 1;
        while (four_mil<10);
    }
    TIM3_CCR2 = 0;
    TIM3_EGR |= 1;
}

void forward_enc(int ticks, int spd)
{
    GPIOC_ODR = 0xF;
    clear_encoders();
    speed(spd);
    while(encoder_avg<ticks)
    encoder_avg = (encoder_bl+encoder_br+encoder_fl+encoder_fr)/4;
    speed(0);
}

void back_enc(int ticks, int spd)
{
    GPIOC_ODR = 0x0;
    clear_encoders();
    speed(spd);
    while(encoder_avg<ticks)
    encoder_avg = (encoder_bl+encoder_br+encoder_fl+encoder_fr)/4;
    speed(0);
}

void right_enc(int ticks, int spd)
{
    GPIOC_ODR = 0x9;
    clear_encoders();
    speed(spd);
    while(encoder_avg<ticks)
    encoder_avg = (encoder_bl+encoder_br+encoder_fl+encoder_fr)/4;
}

void left_enc(int ticks, int spd)
{
    GPIOC_ODR = 0x6;
    clear_encoders();
    speed(spd);
    while(encoder_avg<ticks)
    encoder_avg = (encoder_bl+encoder_br+encoder_fl+encoder_fr)/4;
}
void move_forward(int stop, char sens)
{
    int diff;
    GPIOC_ODR = 0xF;    //All wheels move forward
    while(1)
    {
        float kd, kt;
        if (sens == 'R')
            dist = ping_FR();
        else if (sens == 'L')
            dist = ping_FL();
        else if (sens == 'B')
        {
            dist = ping_BR();
        }
        diff = dist-stop;
        if (diff>400 && (sens == 'R' || sens == 'L'))
            kd = .2;
        else if (diff<400 && sens == 'B')
            kd = .2;
        else
            kd = .25;
        if (dist<stop && (sens == 'R' || sens == 'L'))
            break;
        if (dist>stop && sens == 'B')
            break;
        if (diff*kd<10)
            speed(15);
        else if (diff*kd>75)
            speed(75);
        else
            speed(diff*kd);
        }
        speed(0);
    }

void move_back(int stop, char sens)
{
    int diff;
    GPIOC_ODR = 0x0;    //All wheels move backwards
    while(1)
    {
        float kd;
        if (sens == 'R')
            dist = ping_BR();
        else if (sens == 'L')
            dist = ping_BL();
        else if (sens == 'F')
            dist = ping_FR();
        diff = dist-stop;
        if (diff>400 && (sens == 'R' || sens == 'L'))
            kd = .2;
        else if (diff<400 && sens == 'B')
            kd = .2;
        else
            kd = .25;
if (dist <= stop) break;
if (diff*kd < 10) speed(15);
else if (diff*kd > 75) speed(75);
else speed(diff*kd);
}
speed(0);
}

void move_right(int stop, char sens)
{
  int diff;
  GPIOC_ODR = 0x9;  //BR-forward, FR-backward, BL-backward, FL-forward
  while(1)
  {
    float kd;
    if (sens == 'F')
      dist = ping_RF();
    else if (sens == 'B')
      dist = ping_RB();
    else if (sens == 'L')
      dist = ping_LF();
    diff = dist-stop;
    if (diff > 400 && (sens == 'F' || sens == 'B'))
      kd = .2;
    else if (diff < 400 && sens == 'L')
      kd = .2;
    else
      kd = .25;
    if (dist <= stop) break;
    if (diff*kd < 10) speed(25);
    else if (diff*kd > 75) speed(75);
    else speed(diff*kd);
    speed(0);
  }
}

void move_left(int stop, char sens)
{
  int diff;
  GPIOC_ODR = 0x6;  //BR-backward, FR-forward, BL-forward, FL-backward
  while(1)
  {
    float kd, kt;
    if (sens == 'F')
      dist = ping_LF();
    else if (sens == 'B')
      dist = ping_LB();
    else if (sens == 'R')
      dist = ping_RF();
    diff = dist-stop;
    if (diff > 400 && (sens == 'F' || sens == 'B'))
      kd = .2;
else if (diff<400 && sens == 'R')
    kd = .2;
else
    kd = .25;
if (dist<stop)
    break;
if (diff*kd<10)
    speed(25);
else if (diff*kd>75)
    speed(75);
else
    speed(diff*kd);
} }
speed(0);
}

void move_cw()
{
    GPIOC_ODR = 0x3;       //BR-backward, FR-backward, BL-forward, FL-forward

    void move_ccw()
    {
        GPIOC_ODR = 0xC;       //BR-forward, FR-forward, BL-backward, FL-backward
    }

    void turn_right()
    {
        GPIOC_ODR = 0xC;
        clear_encoders();
        speed(75);
        while(encoder_avg<3925)
        {
            encoder_avg = (encoder_bl+encoder_br+encoder_fl+encoder_fr)/4;
            speed(0);
        }
    }

    void turn_left()
    {
        GPIOC_ODR = 0x3;
        clear_encoders();
        speed(75);
        while(encoder_avg<3925)
        {
            encoder_avg = (encoder_bl+encoder_br+encoder_fl+encoder_fr)/4;
            speed(0);
        }
    }

    int ping_RF(void)
    {
        int dec=0;
        print("1");
        while(newData==0);
        for(int j=0;j<length;j++)
        {
            dec = dec*10 + (readMsg[j] - '0');
            //readMsg[j] = '\0';
        }
        dec = dec/10 + 20;
        newData = 0;
        return dec;
    }
int ping_RB(void)
{
    int dec=0;
    print("2");
    while(newData==0);
    for(int j=0;j<length;j++)
    {
        dec = dec*10 + (readMsg[j] - '0');
    }
    dec = dec/10;
    newData = 0;
    return dec;
}

int ping_LF(void)
{
    int dec=0;
    print("3");
    while(newData==0);
    for(int j=0;j<length;j++)
    {
        dec = dec*10 + (readMsg[j] - '0');
    }
    dec = dec/10 + 20;
    newData = 0;
    return dec;
}

int ping_LB(void)
{
    int dec=0;
    print("4");
    while(newData==0);
    for(int j=0;j<length;j++)
    {
        dec = dec*10 + (readMsg[j] - '0');
    }
    dec = dec/10;
    newData = 0;
    return dec;
}

int ping_FR(void)
{
    int dec=0;
    print("5");
    while(newData==0);
    for(int j=0;j<length;j++)
    {
    }
int ping_FL(void)
{
    int dec=0;
    print("6");
    while(newData==0);
    for(int j=0;j<length;j++)
    {
        dec = dec*10 + (readMsg[j] - '0');
    }
    dec = dec/10 + 15;
    newData = 0;
    return dec;
}

int ping_BR(void)
{
    int dec=0;
    print("7");
    while(newData==0);
    for(int j=0;j<length;j++)
    {
        dec = dec*10 + (readMsg[j] - '0');
    }
    dec = dec/10;
    newData = 0;
    return dec;
}

int ping_BL(void)
{
    int dec=0;
    print("8");
    while(newData==0);
    for(int j=0;j<length;j++)
    {
        dec = dec*10 + (readMsg[j] - '0');
    }
    dec = dec/10 + 15;
    newData = 0;
    return dec;
}
void USART6_IRQHandler(void)
{
    static int i = 0;
    unsigned char temp = 0;
    temp = USART6_DR;
    readMsg[i] = temp;
    i++;
    if(temp == '$' || i >= 5)
    {
        length = i;
        i=0;
        newData = 1;
    }
}

void level_right()
{
    //int dist1, dist2;
    int integral, deriv, error, last_error;
    int spd=1;
    int delta;
    while(spd!=0)
    {
        dist1 = ping_RF();
        dist2 = ping_RB();
        error = dist1 - dist2;
        spd = .1*error;

        if (spd < 0 && spd > -lower)
        spd = -8;
        if (spd > 0 && spd < lower)
        spd = 8;

        if (error < -5)
        {
            move_cw();
            speed(-spd);
        } else if (error > -1)
        {
            move_ccw();
            speed(spd);
        } else
        {
            spd = 0;
            speed(spd);
        }
}

void level_left()
{
    //int dist1, dist2;
    int integral, deriv, error, last_error;
    int spd=1;
    int delta;
    while(spd!=0)
dist1 = ping_LF();

dist2 = ping_LB();

error = dist1 - dist2;

spd = .4 * error;

if (spd < 0 && spd > lower)
    spd = -8;
if (spd > 0 && spd < lower)
    spd = 8;

void level_front()
{
    //int dist1, dist2;
    int integral, deriv, error, last_error;
    int spd=1, 1172;  
    int delta;
    while(spd!=0)
    {
        dist1 = ping_FR();
        dist2 = ping_FL();
        error = dist1 - dist2;
        spd = .4 * error;

        if (spd < 0 && spd > -lower)
            spd = -8;
        if (spd > 0 && spd < lower)
            spd = 8;
    }
    if (error < -3)
    {
        move_ccw();
        speed(-spd);
    }
    else if (error > 3)
    {
        move_cw();
        speed(spd);
    }
    else
    {
        spd = 0;
        speed(spd);
    }
}

if (error < -3)
{
    move_ccw();
    speed(-spd);
}
else if (error > 3)
{
    move_cw();
    speed(spd);
}
else
{
    spd = 0;
    speed(spd);
}
void speed(spd);

else {
    spd = 0;
    speed(spd);
}

void level_back()
{
    //int dist1, dist2;
    int integral, deriv, error, last_error;
    int spd=1; 1209     int delta;
    while(spd!=0)
    {
        dist1 = ping_BR();
        dist2 = ping_BL();
        error = dist1 - dist2 + 25;
        spd = .4*error;
    }

    if (spd < 0 && spd > -lower)
        spd = -8;
    if (spd > 0 && spd < lower)
        spd = 8;

    if (error < -3)
    {
        move_cw();
        speed(-spd);
    }
    else if (error > 3)
    {
        move_ccw();
        speed(spd);
    }
    else
    {
        spd = 0;
        speed(spd);
    }
}

void center(char sens)
{
    //int dist1, dist2;
    int integral, deriv, error, last_error;
    int spd=1; 1249     int delta;
    while(spd!=0)
    {
        if (sens == 'F')
            dist1 = ping_RF();
        }
    }
}
dist2 = ping_LF();
}  
else if (sens == 'B'){
  dist1 = ping_RB();
  dist2 = ping_LB();
}
error = dist1 - dist2 + 200;
spd = .2*error;

if (spd < 0 && spd > -12)
spd = -12;
else if (spd > 0 && spd < 12)
spd = 12;

if (spd > 50)
spd = 50;
else if (spd < -50)
spd = -50;

if (error < -2)
{
  GPIOC_ODR = 0x6;
  speed(-spd);
}  
else if (error > 2)
{
  GPIOC_ODR = 0x9;
  speed(spd);
}  
else
{
  spd = 0;
  speed(spd);
}

void speed(int percent)
{
  flag = 0;
  int speed;
  speed = percent * 640;
  TIM4_CCR1 = speed;
  TIM4_CCR2 = speed;
  TIM4_CCR3 = speed;
  TIM4_CCR4 = speed;
  while(flag==0);
  //TIM4_EGR |= 1;
}

void print (char* msg)
{
  unsigned char i = 0;
  while (msg[i] != 0) // loops through string until there is a 0
while ((USART6_SR & 0x80) == 0); // waits until the previous message has sent
USART6_DR = msg[i]; // send the next char in the string
i++;
}

void printLCD (char* msgLCD)
{
    unsigned char i = 0;
    while (msgLCD[i] != 0) // loops through string until there is a 0
    {
        while ((USART1_SR & 0x80) == 0); // waits until the previous message has sent
        USART1_DR = msgLCD[i]; // send the next char in the string
        i++;
    }
}

void EXTI0_IRQHandler(void) //interrupt ext line 0
{
    encoder_bl++; 
    EXTI_PR |= 1; //clear flag
}

void EXTI1_IRQHandler(void) //interrupt ext line 1
{
    encoder_fl++; 
    EXTI_PR |= (1 << 1); //clear flag
}

void EXTI2_IRQHandler(void) //interrupt ext line 2
{
    encoder_br++; 
    EXTI_PR |= (1 << 2); //clear flag
}

void EXTI3_IRQHandler(void) //interrupt ext line 3
{
    encoder_fr++; 
    EXTI_PR |= (1 << 3); //clear flag
}

void EXTI4_IRQHandler(void) //interrupt ext line 4
{
    encoder_flag++; 
    EXTI_PR |= (1 << 4); //clear flag
}
void EXTI9_5_IRQHandler(void)  //interrupt ext line 6
{
edge++;
EXTI_PR |= (1 << 6);  //clear flag
}

void TIM4_IRQHandler()  //4ms timer interrupt
{
four_mil++;
TIM4_EGR |= 1;
TIM4_SR &= 0xFFFF;
TIM4_CR1 |= 1;  //Resets the interrupt
flag = 1;
}

void TIM3_IRQHandler()  //4ms timer interrupt
{
TIM3_EGR |= 1;
TIM3_SR &= 0xFFFF;
TIM3_CR1 |= 1;  //Resets the interrupt
}

void TIM2_IRQHandler()  //1ms timer interrupt
{
one_mil++;
TIM2_EGR |= 1;
TIM2_SR &= 0xFFFF;
TIM2_CR1 |= 1;  //Resets the interrupt
}