Air to Ground Maze Solver

Hamza Nawaz, Jerrod Rout, Nate Jackson, Will Isidort

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

***Abstract* — Using a quadcopter with an attached camera, a video stream will be sent to a processing hub for image processing which will use maze a solving algorithm to generate a list of commands that will be wirelessly transmitted to a ground vehicle situated in front of a reconfigurable maze. The ground vehicle will use these commands in order to navigate through the maze by using embedded software on an Atmega328p chip to interpret those command and to also avoid collision with the maze walls.**

I. Introduction

With the advent and subsequent popularity growth of UAVs (unmanned air vehicles) and autonomous vehicles, we have begun to see their use and functionality expand and diversify in both civilian and military applications. Piggybacking on this technology boom, we have decided to explore ways in which UAVs and wheeled robots might be implemented to work in concert in a semi-autonomous, Internet of Things type of application in an effort to aid ground personnel in high-risk scenarios. Military departments and public safety organizations with Search & Rescue or Search & Destroy type needs could benefit from the added efficiency and reduced manpower facilitated by such technology.

For our project, we decided to incorporate aspects of robotics, communications, computer vision, and UAV technology by designing a ground vehicle that is capable of navigating a maze based on images taken from a quadcopter positioned above. This will be done by using computer vision techniques to generate a binary image that can be solved through algorithms such as Breadth-First Search and A*. Once a solution is obtained, it will be translated into navigational cues that can be sent to the ground vehicle. The ground vehicle will interpret these commands by using a pre-programmed MCU and onboard sensors such as ultrasonics and rotary encoders. It will continue to traverse the maze until it locates an object placed within (such as a tennis ball) and then exit. The maze itself will be constructed to have a braid-type layout; this will add another dimension to the project by requiring not only a solution to the maze to be obtained but also for the computed path to be the shortest.

This project is a culmination of our research into the various components and concepts needed to realize this design. Hardware will be constructed based on several aspects such as, component cost, power consumption, transmission rate, effective range, resolution, and efficiency. Likewise, algorithms and techniques will be chosen based on ease of implementation, effectiveness, and computation time. A PCB will be designed for the ground vehicle that allows the selected hardware to communicate with the programmed MCU. Once the PCB has been assembled and programs have been written for image processing, maze solving, and navigation, a prototype of the system will be built.

II. System Components

There are several different overall system components that will be involved to get our ground vehicle through the maze successfully including both hardware (ground vehicle design, quadcopter, wireless communication system, wireless video transmission) and software (image processing, maze solving, and embedded programming). Each system component in the flow relies on the previous component to operate successfully, so each component is integral to the overall design, no matter how complex or simple it may be. An overall block diagram of the system can be seen below in figure 1.

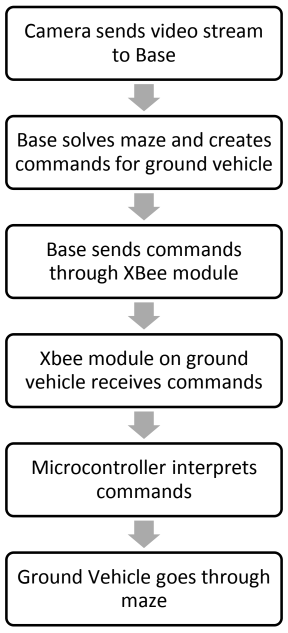

Camera sends video stream to Base
Base solves maze and creates commands for ground vehicle
Base sends commands through XBee module
Xbee module on ground vehicle receives commands
Microcontroller interprets commands
Ground Vehicle goes through maze

Figure 1 Overall Design Flow

A. *Quadcopter & Video Transmission*

The first component in the flow of the overall design is the quadcopter and camera. We chose the DIY Quad from 3DRobotics mainly because it was generously provided to the group by Dr. Richie. Attached to the quadcopter will be a camera that will stream video to a computer which has a receiver attached to it. In order for the video stream to be processed correctly in software, a high quality camera is required, so a GoPro Hero3 camera which is capable of transmitting live standard video was purchased which also has a built-in video transmitter and battery, which also saves on cost and weight, because we want the quadcopter to be as light as possible.

B. *Ground Vehicle*

The ground vehicle we chose was the Pirate 4WD Mobile Platform, which was also generously given to us by Dr. Richie. This was an optimal choice because the platform was designed to mate with Arduino development boards and has locations to put onboard sensors which allowed us to create a working prototype. The ground vehicle has 4 DC motors, which allows us to have accurate in-place turning, and has a speed of 90cm/s. This platform is also small enough so a reasonably complex maze can be built without having to worry about fitting it to the size of our ground vehicle.

C. *Microcontroller*

After researching extensively on which microcontroller would be best suited for our purposes, we decided to go to with the Atmega328p. We chose this microcontroller because it has more than enough digital I/O and analog I/O, and also has 32kB flash memory with read-while-write capabilities. It also has an operating frequency of 20MHz which will allow us to have fast processing time which is needed for our robotics application. Another reason we chose this is has many different open source libraries and support for robotics. It uses the Arduino environment which uses C code, which makes it easier to code as the group is familiar with it. And lastly the cost of these microcontrollers are very low.

D. *PeripheralComponents*

There are peripheral components and sensors that must be added to our ground vehicle if it is to get through the maze successfully. We must attach ultrasonic sonic sensors to our ground vehicle for collision detection, and add wheel encoders to accurately measure how far the ground vehicle has traveled. We chose the HC-SR04 Distance Sensor due to its supported library in the Arduino environment and its accuracy of .3cm, and its more than enough of range of 400cm. For our wheel encoders, the wheel encoder kit from Sparkfun was purchased. After testing these encoders we discovered they had 8 degrees of freedom, but that was accurate enough for our design specifications.

E. *Wireless Communications*

The wireless communications between the computer that solves the maze and generates commands (in a form of a string), and to the ground vehicle is a very crucial part of the design. It was decided that best solution to effectively send these commands was to use two XBee modules (using ZigBee protocol), one connected to the computer, and one connected to the microcontroller to the ground vehicle. The XBee modules were chosen for their overall low power consumption, throughput capacity, long range, and relatively small size. A note should be made that a USB adapter is required to use the XBee module with a computer.

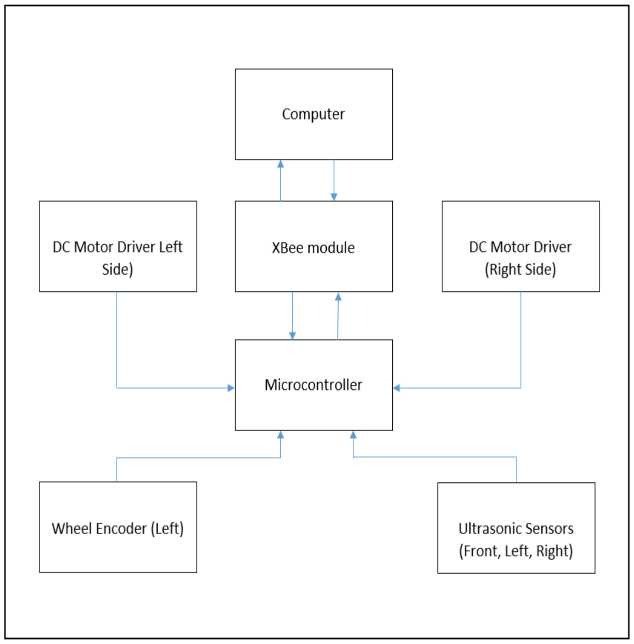
III. Hardware Design and Implementation

The quadcopter is the first part of the design that will operate. In order to safely fly the quadcopter we needed to incorporate a gyroscope, accelerometer, compass, and GPS to the flight controller. We will be giving the quadcopter a pre-programmed mission using the provided Mission Planner software so there won't be any need to manually fly the quadcopter once we start the demonstration. Safety is also a factor we have to consider, so we implemented a geofence so if the quadcopter goes out of range, it will safely land instead of continuously flying. We also needed a flight time of at least 30 minutes so we decided to use a 11.7V 5100 mA/h battery to power the quadcopter.

The overall system flow of the design can be seen in figure 1. It can be seen that overall flow is linear, and each process requires the results of the previous process in order to operate correctly, which does pose risk in the sense of if one step fails, it can cause failure in the entire system.Once the commands reaches our ground vehicle, the microcontroller on the ground vehicle will use the embedded software to interpret these commands. The three main components attached to the MCU are the ultrasonics sesnsors, wheel encoders, and the XBee

Module. The block diagram below in figure 2 represents how the microcontroller communicates with the overall system.

From the block diagram you can see that XBee module is receiving and also transmitting data back to the computer. This data being transmitted is the ultrasonic sensor data which sends the distance to the left, right, and front walls in centimeters, and also the wheel encoder data so we can see on our side that all peripheral components are operating correctly. There are four DC motors on the ground vehicle, but we have tied the left two motors' enable and control pins together so they can operate simultaneously, and also to save on digital I/O pins. We did this for the right two DC motors as well, so one line of code would be able to operate either side of the vehicle. We attached a wheel encoder to the two front DC motors of the ground vehicle, but we realized we actually only needed one encoder, because no matter the type of movement (forward, left, right), every wheel will be turning, so we were able to calibrate the turns off just one wheel encoder.


Computer
DC Motor Driver Left Side)
XBee module
DC Motor Driver (Right Side)
Microcontroller
Wheel Encoder (Left)
Ultrasonic Sensors (Front, Left, Right)

The embedded programming was one the most challenging parts of the overall design mainly due to the constant tweaking and changing required when prototyping and calibrating.With a preset size of the maze in mind, we are able to send commands to the ground vehicle in the form of letters and numbers. If "F, 20" is sent, where the first letter is the command, and second is a number in the unit of pixels. The "20" is converted to a distance in cm and the robot interprets this as "Go Forward, X cm". Using this information we program the wheel encoders to travel this distance because we know the distance of one revolution of a wheel.

Using libraries for the ultrasonic sensors, we're able to convert the pings we receive into a distance to centimeters. In order to effectively use the ultrasonic sensors in the code, we have to constantly check the distance while the ground vehicle is moving, instead of checking the distance after movement. In order achieve this we implemented interrupts into our embedded code which is always running in the background. It uses the encoder pin input which always reads a HIGH or LOW depending on the eight possible states of the wheel encoder. Whenever there is a state change, the interrupt code will be activated and will use the ultrasonic sensors to determine if the robot is too close to the walls. If it is, it will correct itself by having one side of the ground vehicle move faster, and the other one slower in order to get itself on a straight path again. If the robot is at safe distance from the wall, the code will go back to where it left off and continue to run. Below in figure 3 you can see a block diagram of the embedded programming flow.

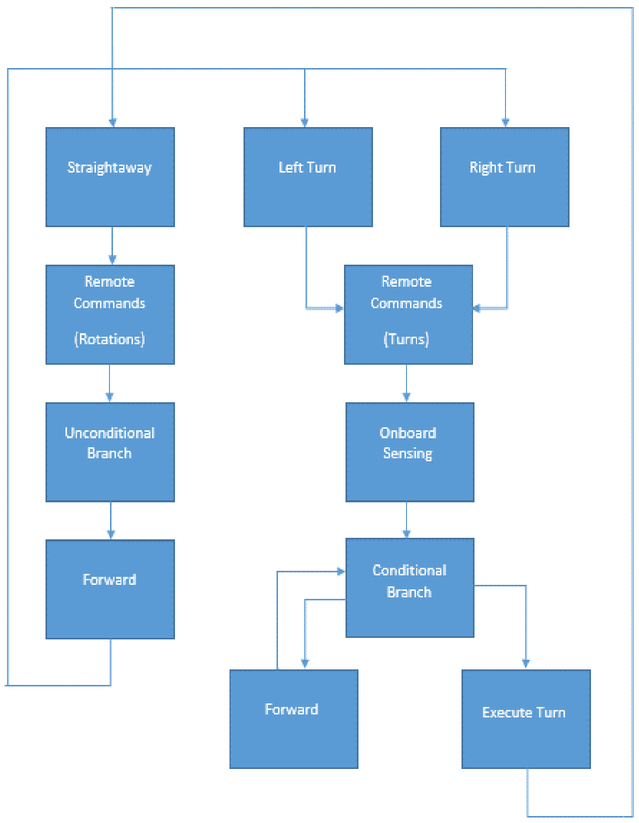


Figure 3 Embedded Programming Diagram

IV. Software Design

We will identify the maze in the image sent by the camera on the quadcopter and solve it through the use of software. The two fields that our project incorporates the most are image processing and graph theory.

A. Image Processing

A substantial part of this project involves analyzing and manipulating images. This will be done with the implementation of the OpenCV image processing library. We will use the following techniques in order to correct distortion, locate the maze, and prepare the image for binarization.

1) Camera Calibration: Prior to being used to detect the maze, the original camera was calibrated to remove distortion by using a chessboard image to develop a camera matrix. However, a higher quality camera (GoPro Hero3) was purchased for the final demo as it gives much less distortion when in video mode. Minimizing distortion is necessary because its presence could reduce the accuracy of the solution by warping the maze walls.

2) Color Thresholding: The boundaries of the maze were found by thresholding for the color blue and analyzing the resulting contours. The largest blue region (contour) is assumed to be the outline of the maze and a bounding box is drawn. The same technique is used to locate the start location (robot position) and goal in the maze. These pixel coordinates are stored for later use.

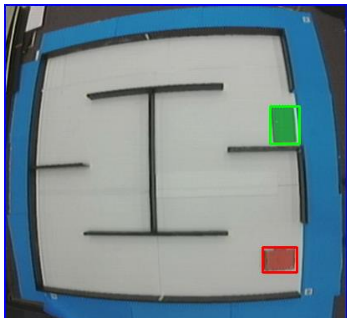


Figure 4 Camera image is thresholded for the colors blue, red, and green to identify the bounds of the maze, robot position, and the goal, respectfully

3) Maze Extraction: The bounding box outlining the maze considers the rotation of the maze and minimizes the area enclosed. This is done to prevent additional background artifacts from appearing in the extracted image. Our program is robust to rotation and will rotate the maze such that it has either a horizontal or vertical orientation. Once positioned correctly, the maze is cropped and extracted.

4) Binarization: Once the maze image has been isolated, binary thresholding is applied to create a black and white image. The black pixels will represent the walls of the maze and white pixels will represent the floor of the maze. The image is also eroded to enhance the accuracy of the solution.

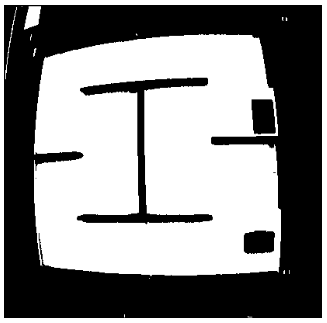


Figure 5 Binary maze which will be converted into a text file and solved

B. Maze Solving

Once a binary image of the maze has been created through image processing techniques, it will be interpreted by utilizing the NetworkX library to construct a graph of interconnecting nodes representing the paths of the maze. Next, an algorithm will iterate through the maze and produce a solution which will be translated and sent to the robot as movement commands. The nature of the BFS algorithm will enable the robot to take the optimal path. The following steps were performed in order to accomplish this.

1) Interpreting Pixel Data: The values of every pixel in the binary image are stored in the form of an iterable list. The list of pixel values is iterated through and a text file is created which represents the layout of the maze. In the text file a pixel value of zero (black, wall) is represented by a '1', a pixel value of 255 (white, floor) is represented by a '0', the start location is represented by an 'S', and the end location is represented by an 'E'. Both the start and end locations are padded with zeros so that the start and goal node are accessible.The created text file is then processed further to reduce the likelihood of false turns (turn commands generated by the meandering nature of the BFS algorithm that are not found in the physical maze layout) being sent to the robot once the solution is obtained.

2) Determining Maze Path Width: A threshold value which differentiates vertical paths from horizontal paths is found by reading the text file created above and counting the number of zeros between two ones. These values are stored in a list and can then be interpreted to find the average path width which will be used to path threshold in when finding Hough lines in the image.

3) Finding Hough Lines: The Hough transform is used to condense the maze paths to one pixel in width. This is accomplished by determining if a detected Hough line is actually a wall in the maze by subtracting neighboring Hough lines of the same orientation and comparing the result to a predetermined path threshold. If the distance between two Hough lines is greater than the path threshold the lines are assumed to form a path and a line is drawn in the middle of them. Once the paths of the maze have been found, the mid-lines are overlaid on the original binary image and this image is subtracted from a solid black image of the same size. The resulting image is a black and white one pixel width line representation of the maze.

4) Connecting Nodes and Solving the Maze: The nodes of the maze will be created and connected by analyzing the text file and interpreting the characters 'S', 'E', '1', and '0'. When the 'S' character is found in the text file the root node is created. The root node 'S' is connected to other nodes by comparing the characters above, below, to the right, and to the left of it. If either of these characters is a '0' the node will be added to a graph and connected. All of the floor nodes ('0') and the goal node ('E') will be linked together in this manner. Once all nodes are connected a BFS algorithm will run and the shortest path connecting 'S' and 'E' nodes will be found. After completion, backtracking will be used to obtain the coordinates of every node in the path.

5)Translating the Solution and Sending Commands: When the solution is obtained by backtracking it is received as a list of nodes named by their coordinates. Their positions relative to one another were translated into cardinal directions and these were then interpreted to generate the forward ('F'), right ('R'), and left ('L') commands. These commands were then filtered according to the size of the maze and distance the robot travels in one revolution. The resulting string was then sent to the robot through serial communications.

V. Maze Layout

The maze is the final part of the project that needs to be addressed. It will be built using black foam boards for the walls and for the floor we'll be using a white surface so we can easily identify the maze in software. In our design, we are planning to use a modifiable maze and also there should be more than one way to reach to the end of the maze so we can choose the most optimal path. All pathways' dimensions will have same size based on the ground vehicle's specific dimension. For instance, the corridors will be reasonably twice as big as the ground vehicle to prevent the vehicle from getting stuck while navigating through the maze. The angle for turning left or right is to be 90 degrees. We will also be able to move the walls so we can test different maze layouts and prove that the design can work with more than one maze layout.

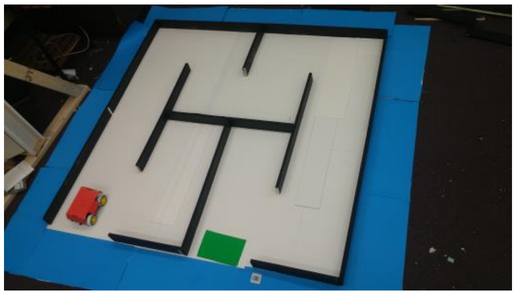


Figure 6 Reconfigurable color-coded maze layout

VII. PCB Design

The main hardware design component in this project came with the implementation of the DFRobot Pirate 4WD robot platform and the design and integration of the PCB used to power it. This is the most important part of our design as it contains all the circuitry needed for the operation of the ground vehicle. Creating a PCB is one of the major requirements for this project provided by the Accreditation Board for Engineering and Technology, so learning how to do it properly was crucial. The ground vehicle PCB is a custom, 2-layer microcontroller that was designed to handle minimal processing, with the bulk of the processing done on the remote processing hub. The Atmel ATmega328p was chosen as the ground vehicle's MCU, given its reliable architecture and its compatibility with the straightforward, easy-to-use Arduino IDE for programming. The ATmega328p also had adequate memory (32KB ISP flash) and clock rate (16MHz, upgradeable to 20MHz) to allow for sufficiently robust and responsive operation of the ground vehicle.

Peripherals operated by the MCU include two H-bridges to drive the DC motors on the ground vehicle, an XBee module to talk to the processing hub, and power regulation circuitry to supply the correct power to all the components. Below in figure 4 you can see the final layout of the PCB and see how everything interconnects. The ultrasonic sensors and wheel encoders were mounted offboard. The MCU I/O to interface with these sensors was broken out from the MCU to SIP sockets, mounted at the periphery of the PCB, to allow for maximum configurability. We decided to wire the ultrasonic sensors to the board instead of placing them directly on the board because they may need to be moved in the future for optimization, and the wheel encoders are attached to the DC motors, so they would not be placed onto the final PCB design (figure below).

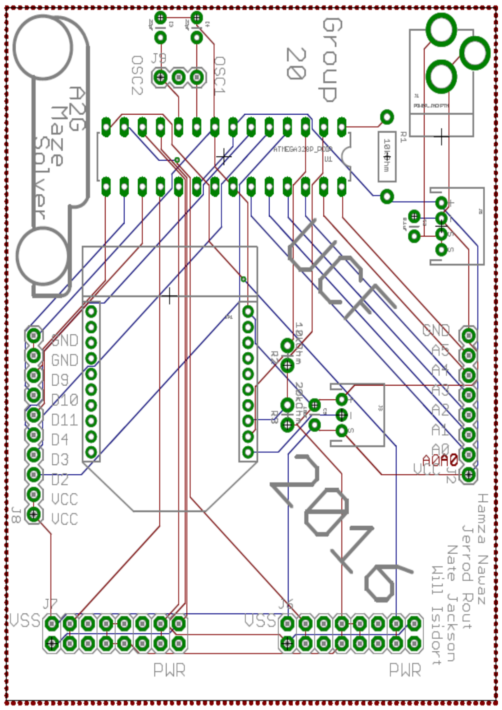
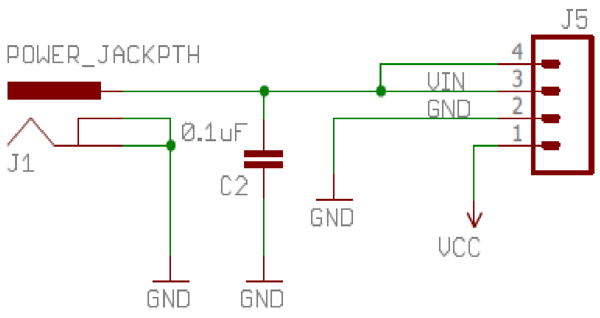
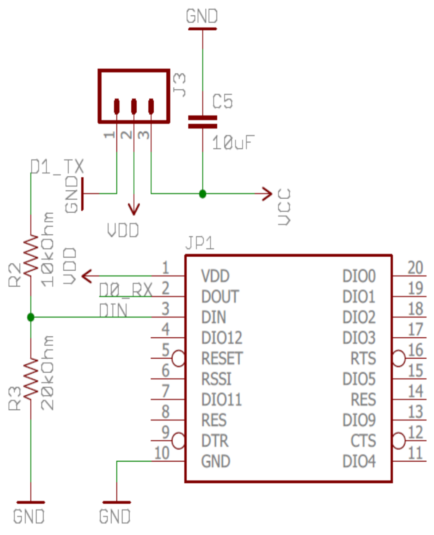


Figure 2 PCB Layout

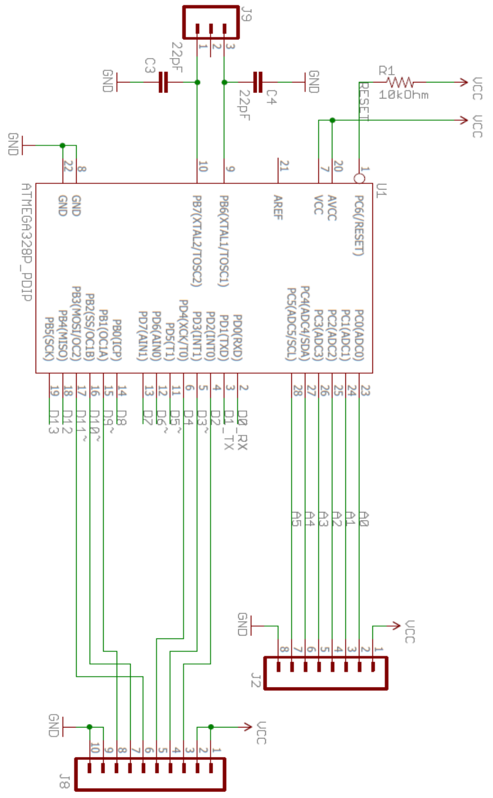
In order to ensure that all the circuitry worked before making and ordering the final design of the PCB, we prototyped the board out onto a breadboard to finalize and validate our design before sending it off to the PCB manufacturer OSH Park.

The ground vehicle has two power supplies: one to power the microcontroller and all peripherals and one to power the DC motors. A 9V cell powers the microcontroller; this is fed into the PCB where it is stepped down to supply a 5V rail to the MCU and all peripherals. The Pololu D24V6F5 5V switching regulator was chosen in order to minimize power consumption. Serial communication is achieved through the onboard XBEE module, which requires 3.3V VCC and input to all pins. To achieve this, a Linear Technology LT1086 linear regulator was chosen to provide a 3.3V rail for the XBEE module. The 3.3V rail feeds incestuously off of the 5V rail, thus a linear regulator was chosen because of their reliability and also because a 1.7V voltage drop would be only slightly above the dropout voltage rating of the LT1086 and therefore power loss would be minimal. The Tx output the ATmega328 is also 5V; a voltage divider circuit was chosen to drop this voltage down to 3.3V since this is not a constant signal (Can be seen in figure 7). Vertical mounted PTH resistors were chosen to help dissipate any heat. The power supply for the DC motors is governed by the H-bridges. We used five 9.5V cells.

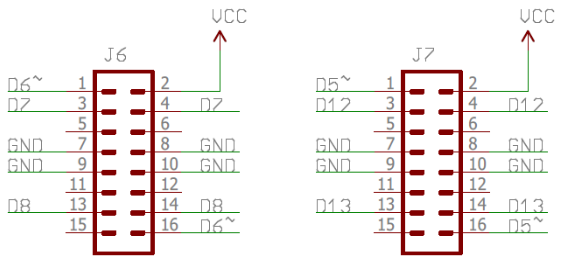




Next we had to design the MCU circuitry and place it in the board layout. The reset pin is required to be connected to our Vcc with a 10kΩ resistor. The MCU also requires a 16 MHz crystal oscillator to operate, which is placed on pins 9 and 10. We use two 22pF capacitors on each pin of the oscillator connected to ground for decoupling and tuning. We put two headers on each side of the board so we can connect our ultrasonic sensors and wheel encoder. This can all be seen in figure 6 below.



The four DC motors each power a separate wheel of the ground vehicle. Control of the DC motors by the MCU is facilitated by two Texas Instruments L293d H-bridge ICs, each controlling one pair of wheels, both on the same side. Although two H-bridges can allow for independent control of up to all four wheels, the control pins of each H-bridge were tied together to the same output pin of the MCU, effectively syncing each pair of motors together. This reduces path deviation, since both motors spin in lockstep with each other, and also reduces the number of occupied GPIO pins of the MCU. A larger PCB layout was implemented, with the H-bridges mounted away from other components, and copper pours were added to the top and bottom layers of the PCB to allow for enhanced heatsinking. Locomotion is achieved through 5V DC input signals to the enable pins (to control wheelspin direction) and a 5V PWM signal to the control pin, with wheelspin speed adjusted by varying the duty cycle.



VIII. Conclusion

Overall this project allowed us to use the concepts and techniques we have learned in our electrical engineering program at UCF. It was a challenging and rewards experience for all four members of the team. We were able academia or in the industry.

We would like to thank and acknowledge Boeing, Leidos, Dr. Richie and Dr. Wei, and the UCF Foundation for providing us with the funding and materials to build the project. We would also like to thank the professors and engineers that have agreed to be a part of our review board committee and took the time to review our project and evaluate it.

Biography

Hamza Nawaz is currently a senior at the University of Central Florida. He is also currently part of the college work experience program working at Lockheed Martin Missiles and Fire Control and has accepted a full time Electrical Engineering position there.

Jerrod Rout is currently a senior at the University of Central Florida. After graduation he plans to pursue a career in Electrical Engineering within the field of microelectronics.

William Isidort is currently a senior at the University of Central Florida. After graduation he plans to join the workforce and later on pursue a master's degree in Electrical Engineering to further his career path.

Nate Jackson is currently a senior at the University of Central Florida. After graduation he plans to continue his education and pursue his master's degree at UCF in Electrical Engineering.

References

[1]"Mission Planner Support | 3DR | Drone & UAV Technology", *3DR | Drone & UAV Technology*, 2016. [Online]. Available: https://3dr.com/kb/mission-planner/. [Accessed: 08- Apr- 2016].

[2]"Arduino - AttachInterrupt", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Reference/AttachInterrupt. [Accessed: 08- Apr- 2016].

[3]"Using EAGLE: Board Layout - learn.sparkfun.com", *Learn.sparkfun.com*, 2016. [Online]. Available: https://learn.sparkfun.com/tutorials/using-eagle-board-layout. [Accessed: 08- Apr- 2016].

[4]"OpenCV-Python Tutorials", *OpenCV Dev Team*, 2016. [Online]. Available: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html [Accessed: 08- Apr- 2016].

[5]"Creating a Graph using NetworkX", *NetworkX Developers*, 2016. [Online]. Available: https://networkx.github.io/documentation/latest/tutorial/index[Accessed: 08- Apr- 2016].