Team Tango

Notre Dame Electrical Engineering
Senior Design 2025-2026

Annie Marquitz, Edgar Rodriguez, Ben Walsh,
Daniel Harrison

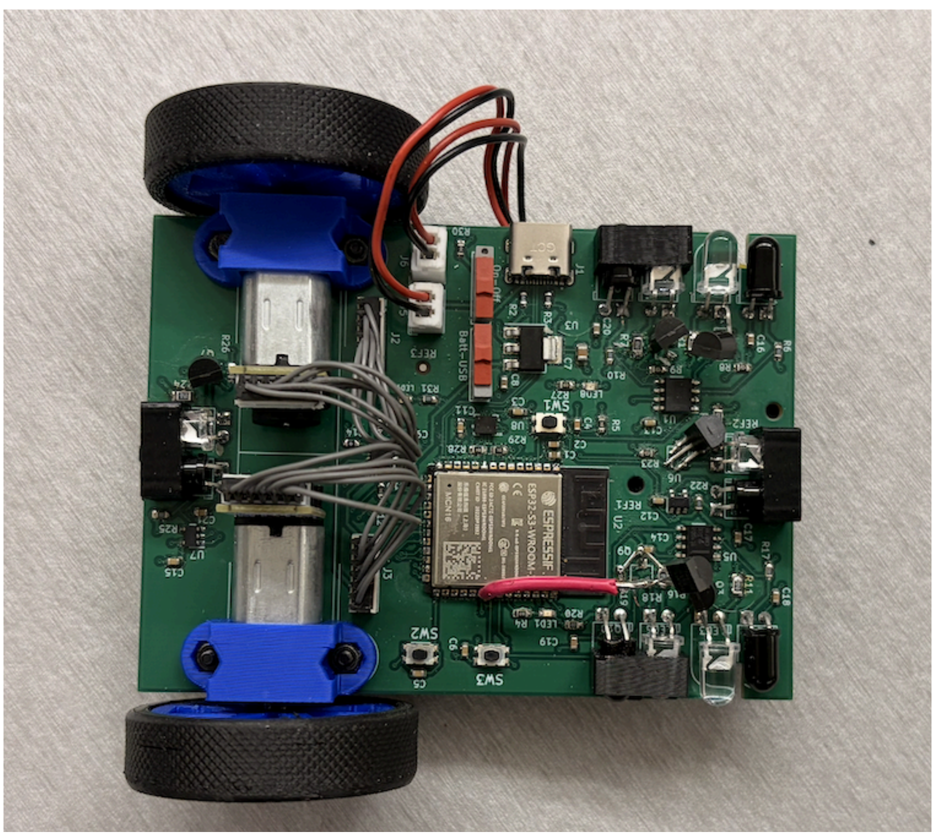


Table of Contents

Introduction

The Team Tango micromouse project was developed as part of a larger effort to create a shared design experience for future EE cohorts. As Professor Schafer prepares for retirement, he proposed that all teams work on the same core project so that the course can better support larger groups of students while still giving everyone a meaningful hands-on engineering challenge. The project is inspired by the traditional Micromouse competition, in which small autonomous robots are designed to navigate and solve a maze without human control. In competition, the maze is arranged as a 16 by 16 grid of cells, and the robot uses sensors, motor control, and path-planning logic to explore the maze and eventually find the fastest route to the goal.

For Team Tango, this project requires combining hardware design, embedded programming, sensing, control, and system integration into one functional robot. Our objective was to create a micromouse capable of detecting maze walls, controlling its motors, maintaining stable movement, and eventually making decisions as it moves through the maze. While the final robot is based on the larger micromouse challenge, the project also served as a practical learning experience in designing a complete autonomous robot from the ground up. As a Senior Design project, it required us to apply and integrate the skills we have developed throughout the past four years. Through this process, our team gained experience with circuit design, sensor integration, motor control, fabrication, debugging, and the challenges of turning an idea into a working robotic platform.

Key System Requirements

The key system requirements centered on controlled movement through the maze and reliable wall detection on all three sides of the mouse. The mouse needed to drive straight, execute accurate 90° and 180° turns, and coordinate both motors independently using encoder feedback for distance and turning control. Power management and motor driver design were important board-level considerations to ensure the motors could operate at the required speeds without affecting the rest of the hardware. The physical size of the mouse was also a requirement, as it needed to be small enough to fit within the maze corridors and complete turns without clipping walls. Our board measured roughly 65.5mm by 90.5mm, which kept the robot within the necessary size constraints for reliable navigation.

For sensing, IR LED and phototransistor pairs were selected because they offered fast sampling rates and consistent accuracy, which were both necessary for making real-time navigation decisions. The sensors needed to detect walls accurately at a minimum of 35mm on the front, since that is the distance at which the mouse stops and aligns to a front wall, and at around 50mm on the sides to stay centered in a corridor. The sensors also needed to update fast enough to match the control loop, which runs every 20ms, so any sensors too slow to keep up would cause the robot to react too late to nearby walls.

Detailed Project Description

Power Planning

The robot is primarily powered by two 3.7V LiPo batteries, which provide a total (nominal) voltage of 7.4V. In reality, these batteries can provide up to 8.4V, so our power regulation system needs to account for this fluctuation in power as the battery charges and discharges.
This 7.4V supply was chosen to give power to our main motors, which are rated up to a maximum of 12V. Based on our search for motors, we found that the two most common options were 6V or 12V. This leaves us with two options: use 7.4V to power the 6V-rated motors by using a regulator to limit the voltage, or use the 7.4V to directly supply the motors without an additional regulator. We chose this second option because although our batteries underpower the motors, preliminary testing confirmed that the motors can easily be operated at this lower voltage without significantly losing out on speed or torque. The 7.4V battery supply provides enough voltage for our robot to move while keeping the system compact and manageable by removing an extra IC from the board.

The rest of the board is powered by 3.3V, giving power to the microcontroller, sensors, and supporting circuitry. We use a low-dropout voltage regulator to ensure the voltage does not rise above 3.3V and burn out any components on our board.

Our board can also be powered by the 5V supply voltage that is delivered by the USB-C connection from a computer. A switch allows us to manually switch the 3.3V power source to come from either the batteries or from the USB port. This allows us to disconnect the batteries for charging or safety and still be able to interface with the microcontroller to upload new code or test the sensing system. For the final autonomous run, this switch will be switched such that all power comes from the batteries.

Similarly, we have a second switch that only switches the battery supply on and off. This switch was important for us to use in our board design to have an emergency off for the LiPo batteries which can present a fire hazard if damaged or misused. Importantly, the motors will only receive power when both this switch allows battery power to be delivered and when the batteries are plugged in to our board, meaning any driving tests must require the batteries to be on the board.

Four-Layer PCB Design

Although a two-layer PCB would have been simpler and less expensive, it was not practical for our design because of the number of components and connections on the board. The PCB needed to connect the microcontroller, motor driver, IMU, sensors, and other components in a compact space. With only two layers, many of these traces would have been difficult to route clearly without crowding the board or creating routing conflicts.

We chose a four-layer design to give ourselves more routing flexibility. All four copper layers were configured as signal layers, allowing traces to move between layers through vias when needed. This made it easier to organize the layout, reduce congestion around dense areas of the board, and keep the overall PCB size manageable while still including all of the required mouse hardware.

Sensing Scheme

The robot uses analog IR sensing to detect walls in the maze. IR LEDs emit infrared light, which reflects off nearby walls and is detected by phototransistors that produce a voltage proportional to the intensity of the received light. A stronger reflection generally indicates that a wall is closer, while a weaker reflection indicates that the wall is farther away.

Our first tests with our board used a different solution with Time-of-Flight (ToF) sensors that integrate the sensing and conversion into a real distance all in the same IC. We ultimately decided to move away from this approach because the ToF sensor involved a serial communication via I2C that introduced long delays into each request for data. From our experimentation, IR sensors could provide a much lower measurement latency making them better suited for the fast, short-range wall detection needed in the maze. By avoiding a reliance on slow I2C communication from the ToF, the analog IR sensors provide a simpler approach with a voltage reading that can be directly polled by the microcontroller. This approach also means that we must build the voltage-to-distance conversion into our software, but this was a relatively trivial task after mapping experimental data to our sensors' readings.

The robot includes two side sensors on each side for helping the robot maintain its centered position within the maze cell, one front sensor for detecting walls ahead of the robot, and one back sensor for providing additional feedback useful for alignment and position correction within the maze. The sensor signals are each processed through an operational amplifier to smooth and condition the output before being read by the microcontroller. This helps make the sensor readings more stable and useful for our PID control to detect walls and make smooth corrections to align our robot.

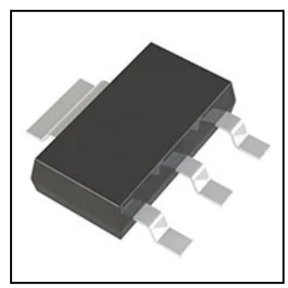
Four of the sensors (one on each side) use a 3D-printed custom shroud to immobilize the LED and phototransistor to the board. This allows the sensors to maintain their calibration over time and through slight bumps to the board. The shroud lets us save lots of time recalibrating our sensors, making our readings more accurate and reliable. Further, the shrouds isolate the IR sensors from each other and from the ambient environment which reduces interference from surrounding sensors or sunlight. However, in our testing of these shrouds we discovered that the shrouded sensors experience a voltage dropout when the robot gets extremely close to the wall, at distances below about 15mm. This makes it look like the wall is farther away when in reality it is dangerously close to the robot. To compensate for this effect, we added the second sensor on each side and left them unshrouded to act as a close-range "bumper" for the walls. This is useful because the unshrouded sensors provide additional close-range detection so the robot can recognize when it is very close to a wall and immediately correct its position.

In addition to the IR sensors, the robot uses an IMU for gyroscopic correction. The IMU helps track changes in heading so the robot can drive straighter and make more consistent turns. Motor encoders are also used to provide feedback on wheel rotation, allowing the robot to estimate driving distance and improve movement accuracy.

Hardware

Each element of our mouse board was selected after informed thought about each solution and most involved some testing to arrive at the final decision. In the following section we will describe the engineering reasons behind selecting each piece:

ESP32-S3-WROOM-1 – The microcontroller for our mouse. The microcontroller receives signals from our sensors, sends signals to our actuators, and stores the control software that operates in our mouse. We chose this model primarily because the Espressif family of chips is very commonly used in development boards and is well documented. Similar boards were available from Prof. Schafer to begin testing immediately that allowed for early prototyping. The S3 has up to 36 GPIO pins which is enough to accommodate all 6 pairs of IR sensors (12), motor controls (6) and encoders (4), USB communication (2), IMU (3), LED (1), and pushbutton (1) for a total of 29 GPIO pins being used. The S3 also has WiFi capabilities that allow the board to report its status in real time as it solves the maze. The microcontroller choice drove a lot of our decisions down the line so crucial for our ultimate design.



AZ1117 Low-Dropout Linear Regulator – The voltage regulator for our 3.3V power planning. The microcontroller needs an input voltage of 3.3V so we made the design decision to run 3.3V across our board and operate all non-motor components on 3.3V as well. The low-dropout is best because it drops the least amount of voltage in order to operate, especially important when its supplied voltage is near 3.3V. We chose this model because it was a stocked part in the EIH and is able to handle the range of voltage and current delivered from both the USB (5V) and from the batteries (8.4V - 7.4V). There was no reason to buy a new component when the stocked part would suffice.

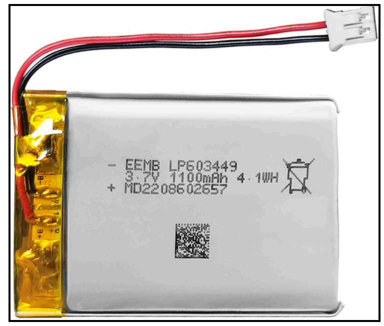
GCT USB4105 Type-C USB Connector – The connection port for USB communication between our personal computers and the microcontroller. The USB also provides 5V power to the board which allows it to be tested even without the batteries. USB-C is a standard connection port for most devices and has plenty of established protocols to communicate with our chosen microcontroller. This would save us from the added difficulty of trying to only communicate via traditional serial like SPI or I2C. We chose this specific model because it was a stocked part in the EIH and has the small form factor appropriate for a micromouse.

C&K KMR2 Microminiature SMT Tactile Switches – The press-buttons on our board to boot and reset the microcontroller. These are necessary for the fundamental operation of the microcontroller and allow the board to easily reset with the loaded program in the maze. These need to be push-button style switches that can handle the 3.3V signals on our board. We chose this model because it was a stocked part in the EIH and is the appropriate size for our micromouse.

WE 450301014042 Mini Slide Switches – The switches that control the power to the whole board from USB and/or battery. Our design has one switch that turns the battery power on/off, and another to switch board power between USB and battery. With this setup the board can be evaluated and programmed without any batteries (e.g. if the batteries are being charged). We only need to connect the batteries when we want to actually turn the motors. We chose slide-switches in this application so that the choice of power stays toggled until it needs to be changed. This is ideal functionality for an on/off switch. We chose this model because it adequately handles the voltages in our board and has a very small form-factor.

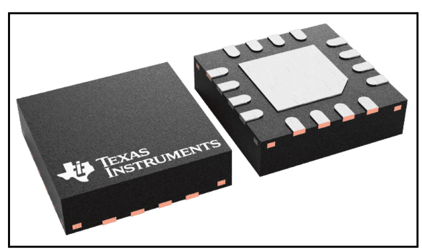
12VDC 500RPM N20 Geared Motor – The drive motors for our wheels. The motors create all the movement for our mouse including driving forward/backward and turning. Our design uses 2 wheels positioned on the back of the mouse. The motors need to have enough torque to move the mouse while also being fast enough to ensure completion of the maze within the 10-minute competition window. We are using the N20 size motor in order to ensure our mouse can fit and turn without hitting the walls in the maze; we found that the motors limit the width of the mouse in our 2-wheel layout. Any measures to shrink further would involve designing a mechanical drive train which we decided would not be appropriate for our first attempt at maze-solving. DC brushed motors are very common for this kind of low-power, high-speed device and there are many available variations of RPM to select from. We decided to use a 12V motor to avoid the extra voltage regulation that would be required to step 7.4V batteries down for a 6V motor. Even though our batteries only provide 7.4V nominal, 12V is the MAXIMUM voltage that the motor should see. In early experimentation we found that the motor functions well even when supplied with this lower-than-rated drive voltage. We chose this RPM because it balances torque with speed in an amount established by our early trials. We tried both 1000rpm and 300rpm motors (the exact same product otherwise) and desired a solution that compromised on speed in order for operation at lower PWM (higher torque). The 300rpm was just too slow. The 1000rpm struggled to turn the motors until almost full PWM, leaving it unable to perform at lower PWM signals. This is especially necessary for our mouse that needs variable-speed motors in order to make slight course adjustments as it drives through a corridor so the 500rpm variation is where we settled. This series of motor models are easily purchased on Amazon which made them a convenient buy for our team.

EEMB LP603449 3.7V LiPo Batteries – The batteries to power the motors. We needed large batteries that are able to give the necessary high current draw from the motors and the easy choice was to use LiPo battery cells. These are well-established to be the best choice for hobbyists due to the high energy density of the LiPo. They are rechargeable and small enough to easily be carried by the micromouse. We decided to pair two of these LiPo batteries in series to give our motors a nominal 7.4V supply: good for powering the 12V motors discussed above. This is just the *nominal* voltage though, so we had to make sure our design could handle the battery voltage ranging from 8.4V (when fully charged) to around 7.4V (when needing charging). We chose this model because it was provided to us by Prof. Schafer, and suits our needs well. It is important to note that LiPo batteries do pose a significant fire risk so we need to be careful not to overcharge, undercharge, short, or puncture the batteries.



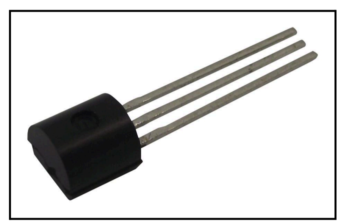
Custom Battery Holder Undercarriage – The holder for our batteries also serves as the bottom of our mouse. We designed this mount to hold our batteries in a discrete out-of-the-way location, allowing interaction with the top of the PCB (pressing buttons/switches, soldering changes, etc). The holder also has a ramp structure built into the front to allow the mouse to traverse small bumps between panels in the maze despite the low clearance.

DRV8411 Dual H-Bridge Motor Driver with Current Regulation – The motor driver for our motors. The motor drivers use transistor switches to enable the motors to receive power as controlled by the microcontroller signals. This is necessary because the microcontroller itself cannot provide enough power for the motors, and we need the motors to have very precise PWM control. We chose this model because it has adequate voltage handling capabilities across our range

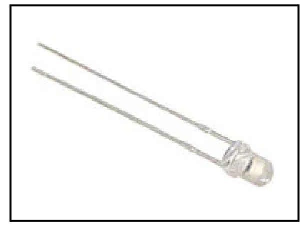


of voltage inputs, and is able to provide enough current through it to give our motors the strength they need to drive. This model conveniently contains 2 H-bridge modules in it so just one IC can control both of our motors. Even more, it has a nice feature in its FAULT pin, which allows us to send a signal back to the microcontroller when things are going wrong and we can quickly cut power to the motors in the interest of safety.

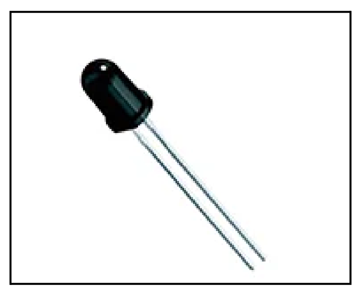
LSM6DSV Inertial Measurement Unit (IMU) – Part of our suite of sensors to keep track of our robot's motion. This sensor uses a gyroscope and accelerometer to track motion along all 3 axes to supplement the data received by the motor encoders and sensors. This information is particularly useful to make higher precision 90° turns in the maze when our control algorithm may not have perfectly straightened the mouse's heading. Especially in cells that do not have walls on the sides to provide the IR sensors with data, we relied on the IMU. We chose this model because it has a small footprint and provides us with the motion sensing capabilities we need. It operates on 3.3V and integrates simply with our board and sensing scheme.



TN0104N3-G N-Channel MOSFET – The switches to turn the IR LEDs on and off. This is needed because the microcontroller pins cannot provide enough current for the LEDs itself so the MOSFET allows the weak control signal to properly actuate the IR light generators. It is necessary to turn the LEDs on and off in order to prevent interference across sensors and save power which would be needlessly spent emitting light. We chose this model because it can easily pass enough drain-source current to power the LEDs and also meets the other requirements for voltage/current maximums. Its threshold voltage of 1.6V is squarely within the maximum voltage able to be supplied by the microcontroller, which makes certain this switch can be functionally operated. In our first attempt at making our board, we discovered that we had messed up the footprint assigned to the **surface-mount** MOSFET part in our PCB design program, and at that point it was too late to order another replacement board. However, with the recommendation of Prof Schafer, we ordered **through-hole** MOSFETs to then attempt to solder straight on to the PCB pads. After receiving the new through-hole parts we were able to creatively make the electrical connections between components necessary to wire the MOSFETs into place on our board.

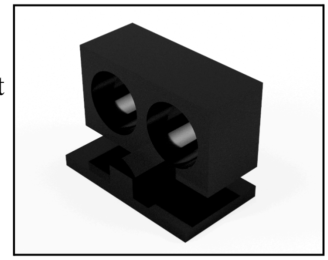


INL-5AMIR15 Infrared Emitter – The IR LEDs used to generate the light that will be detected by the phototransistors. After transmission, the light will interact with the maze walls and reflect back towards the board, where it will be sensed by the phototransistors (discussed below). We chose this model because it provides a spectrum of IR light that matches with the wavelengths to which our phototransistor is sensitive: nominally 880nm. This LED fits within the current and voltage ratings needed for our application and has a very narrow cone of emission allowing for a more precise collimated beam. We chose these through-hole components for the additional structural faculties that we made use of by bending the leads at 90° angles to point them away from the board. This let us avoid the unnecessary production of an auxiliary perpendicular board, and meshed well with our development of the custom sensor shrouds to hold the LEDs and phototransistors in place.

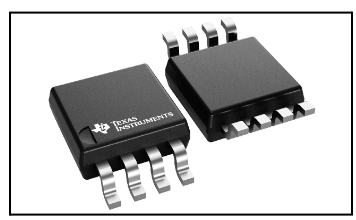


SFH 313 FA-3/4 Phototransistors – The light-sensitive sensors that detect the walls' proximity. These devices generate a voltage proportional to the intensity of incident IR light. When the wall is near, most of the light from the IR LED will be reflected right back onto the phototransistor; but when the wall is far away, light will be dissipated and radiate enough to provide a significantly lower intensity. The microcontroller is then able to poll the pin connected to the phototransistor and immediately read how close or far the wall is. This is crucial to our micromouse because hitting a wall almost always kills a potential maze-solving attempt. When the mouse is driving down a hallway, the sensing mechanism must be fast and accurate enough to quickly correct for error in the centering of the mouse.

We chose this model because its absolute maximum ratings fit within our application and its spectral range of sensitivity matches very closely with the IR spectrum generated by our IR LEDs. Similarly to the LEDs, the through-hole mounting technology allows us to bend the leads to angle the device perpendicular to the PCB and fit in the custom sensor shrouds we designed.



Custom Sensor Shrouds – Shielding casings for the IR LED/phototransistor pairs. The shrouds' primary function is to immobilize the sensors from any movements that could cause the calibration of our sensors to be thrown off. Even a slight bump was causing us to have to recalibrate the received voltage from the phototransistor to its corresponding distance, but the shrouds prevented any changes in the positions of the IR generators and sensors. Also, the shrouds importantly isolate the sensors from each other and the ambient environment. The shroud limits the phototransistor to only detect light incident from the direction it is pointing, reducing interference from the other sensors and from ambient sunlight shining through the Stinson-Remick windows. We designed the shroud to snugly fit the phototransistor and LED, and to mesh into slots we designed as a part of our PCB. The color of the 3D print was crucial to be black so that light would be absorbed and not continue to be reflected around inside the shroud. We went through many trials of varying thicknesses, spacings and angles before setting on our final design that shrouds our sensors the best.



TLV9002 and TLV9001 Operational Amplifiers – Op amps for analog signal processing from the IR sensors. These smooth the voltage signal from the phototransistor so that our microcontroller gets a more useful and less noisy communication. The 9002 model is a dual op amp package which works nicely for the right and left sides because they have 2 sets of sensors immediately adjacent. The front and back, which only have 1 IR sensor each use the smaller 9001 model. We chose these models because they fit our application's current/voltage requirements and provide the op amp functionality in a small package.

Software

The software component of this project was facilitated through VS code, using C++ programming language. The software is made up primarily of 8 .cpp files along with their respective .h files. There is one main.cpp file. There is also a pins_arduino.h file in the workspace that defines all of the pins, which are specific to our board.

The .cpp files include:

- Imu.cpp
- Encoders.cpp
- Main.cpp
- Maze.cpp
- Motors.cpp
- Planner.cpp
- Sensor_calibration.cpp
- Sensors.cpp
- Calibration.cpp

The control algorithm is facilitated by a PID controller. Corrections are implemented as variations in the motor PWM during drive segments. These corrections could be sourced from several sources of feedback. The most significant of these are the IR sensors which are used in the two wall and one wall case with the same PID coefficients. In the two wall case the target is for the distance to each wall to be equal with a deadband of 4mm to avoid over active corrections when unnecessary. In the one wall case the target is 50mm on the side where the wall still is and it uses that 50mm as a reference in the same way that it would use the other wall measurement. Whether there is a wall or not is determined based on a threshold value of 100mm. In the no wall case the sensors are not used, instead the heading is corrected for using the IMU. This is a much slower source of feedback, only running at 60Hz instead of the IR sensor's 330Hz, but it is valuable when the sensors have no wall in range. For all modes some correction is given by the encoders. These have much more influence in the no wall case than in either the one or two wall cases.

The sensors are calibrated by recording the raw sensor values at a series of distances from a wall and fit to a function. Best results were seen by using a piecewise hyperbolic. These calibrations were stable due to the use of the sensor shrouds for the front, back, and one of each of the side sensors. The front side sensors were used as “bumpers” to account for the drop off in seen reflections at very close ranges. These sensors each had a tuned threshold that when the value was greater than would override all other sensors and apply a maximum correction away from the wall.

For turning there are two types implemented. The first is a non-verified turn. This turn runs the motors until a target encoder differential is reached, at which point it applies brakes to the motors and continues driving forward. This type of turn is sufficient when both the current and destination cell have at least one wall. The second type of turn is the verified turn. This turn operates the same as the non-verified; however at the end it will correct its heading until the IMU indicates that it is within one degree of the target cardinal direction. This correction is applied with short pulses of the motors until it is on target.

There are two main modes that the mouse has. First the exploration mode. In this mode, the mouse is determining where walls exist in the maze, and saving that. During the exploration mode verified turns are used to straighten because the mouse does not know if there is a wall to correct to in the destination cell. Exploration mode is used whenever no wall is known on the destination cell. The second mode that the mouse has is a fast mode. This is activated whenever the cell has been visited before and is known to include at least one wall to correct the orientation. In this mode the verified turns are skipped for known safe cells, pauses between actions are eliminated, and base PWM speed is increased for any "straights" defined as two or more drive one cell actions consecutively.

The maze search algorithm is floodfill. A floodfill is an optimistic maze solve algorithm where it is always assumed that no walls exist unless it has been previously mapped. This means if a faster route is possible it will be checked by the mouse before it ever repeats the same route again. The way that our program specifically runs is the mouse completes a floodfill to the goal, then completes the floodfill again on the way back to the start (but on a different route if possible), it will then repeat this process until it has found the fastest path to the center. Then it will make its way back to the starting point of the maze, enter 'fast mode' and follow the fastest path that it has identified. Some testing was done on a weighted Dijkstra algorithm that would have prioritized straights over turns but difficulties with mapping in this new configuration lead to the reversion to the basic flood fill.

System Testing

System testing came in a number of ways, starting with the provided practice mouse and evolving as we made our own board design and software. Initial testing utilized the given practice mouse. Having access to this practice mouse allowed for a quick and accurate account of how the micromouse would move.

First, we tested the motors. Initially we used 200 rpm motors on the practice mouse. We determined that these motors were too slow, and we could gain more speed. The next motors we tested were 1000 rpm. These were determined to be too fast for testing purposes. We settled on 500 rpm motors. These motors were determined to provide enough speed, but also be slow enough for testing at lower PWMs.

The next system component that was tested utilizing the practice mouse were the sensors. First, we tested the Time of Flight sensors. These sensors worked well and we were pretty happy with their performance. However, after some consideration, we decided to test analog IR sensors as well. This LED/phototransistor combination allowed for a very quick sampling period. They also did not involve an additional breakout board, and were easy to work with. This testing allowed us to see that the IR sensors were a smart choice.

The control system of the micromouse was one of the most difficult challenges we faced. This is probably the aspect that took the most amount of testing. First, the control system was tested on the practice mouse. The several Design Reviews completed throughout the semester helped with getting this process started. Initial testing had a goal of getting the mouse to drive straight, when there were no walls present. This testing process included obtaining desired encoder counts for driving a certain distance straight. This value was tweaked many times. It also involved tuning PWM and the PID control parameters. Much of this testing came by analyzing the mouse’s movement with certain parameters, changing one parameter at a time, and evaluating ideal performance before moving to the next parameter. Then, the S-maze was an excellent challenge of the mouse’s controls. This allowed for even more testing of how the mouse would navigate a maze successfully.

Then, the control system had to be implemented onto our own micromouse. Previous research suggested that an IMU would be a helpful aspect in dealing with controlling the mouse, and therefore was added to the new mouse. This added another aspect of testing the controls on the new mouse. The testing process was similar to that of the practice mouse. Much of what was applied to the practice mouse was also applied to the new mouse. We started by getting the mouse to drive in a straight line, with no wall constraints. Then added turns, making sure they were accurate by using the IMU features. After these system components were tested, we increased speed. The testing of speed followed the same patterns described above.

Overall, system testing followed a strong trial and error pattern. We often found that changing one parameter had an effect on multiple other behaviors and aspects of the mouse. For example, when PWM was increased for quicker solving, the PID parameters were not as accurate. Testing was an intensive and important process in building an efficient mouse.

Conclusions

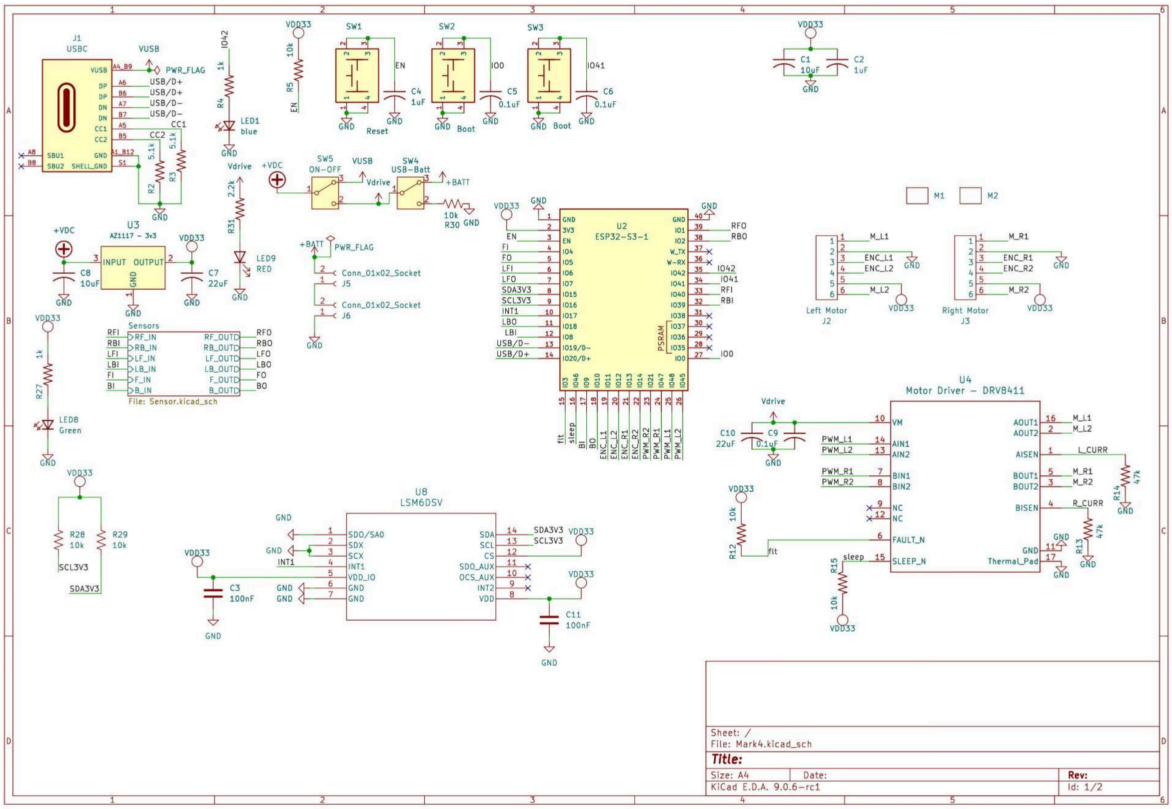
Throughout the past 2 semesters, Team Tango successfully engaged in and completed the design of a micromouse that could solve a maze. On test day, our mouse successfully solved the given maze in 1 minute and 38 seconds. It completed the exploration and fast run in under the allotted 10 minute time period.

As a group, we have discussed things we would have done differently if we were able to start this project over again. The major changes we would have made include:

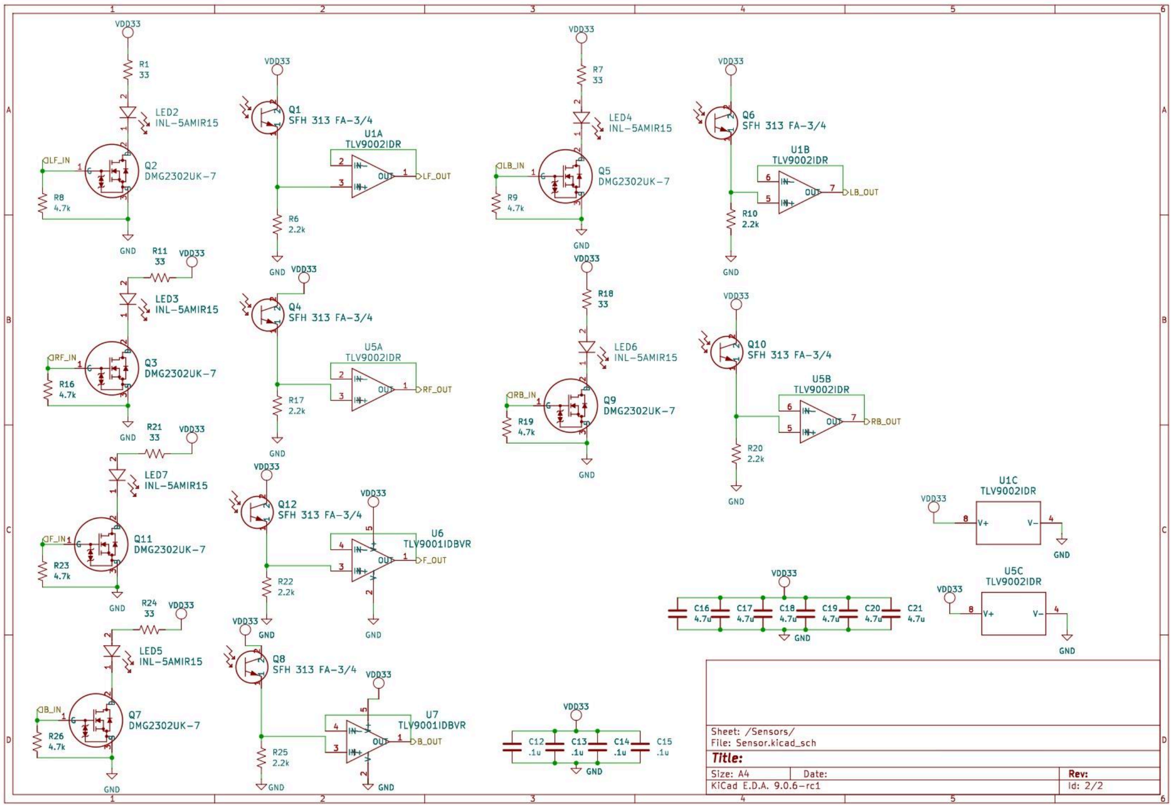
- Making the board smaller, which would have made the mouse itself smaller, allowing for a smaller risk of hitting walls.
- We placed a sensor on the back of the mouse, but don't use it very much. In the future we could change our sensor setup to better sense the maze around us without wasted space.
- There is of course potential to get this mouse moving quicker. We could potentially increase motor rpm if the control system allowed it.
- As far as project management, we would have ordered our mouse board sooner in the semester. This would have allowed us to have even more time to customize the control system for our mouse specifically.

This senior design project allowed us to not only challenge ourselves by implementing knowledge we have gained in past electrical engineering coursework, but also allowed us to improve our project management skills. Technical knowledge, teamwork, time management, and budget management were all skills that were improved upon during this process.

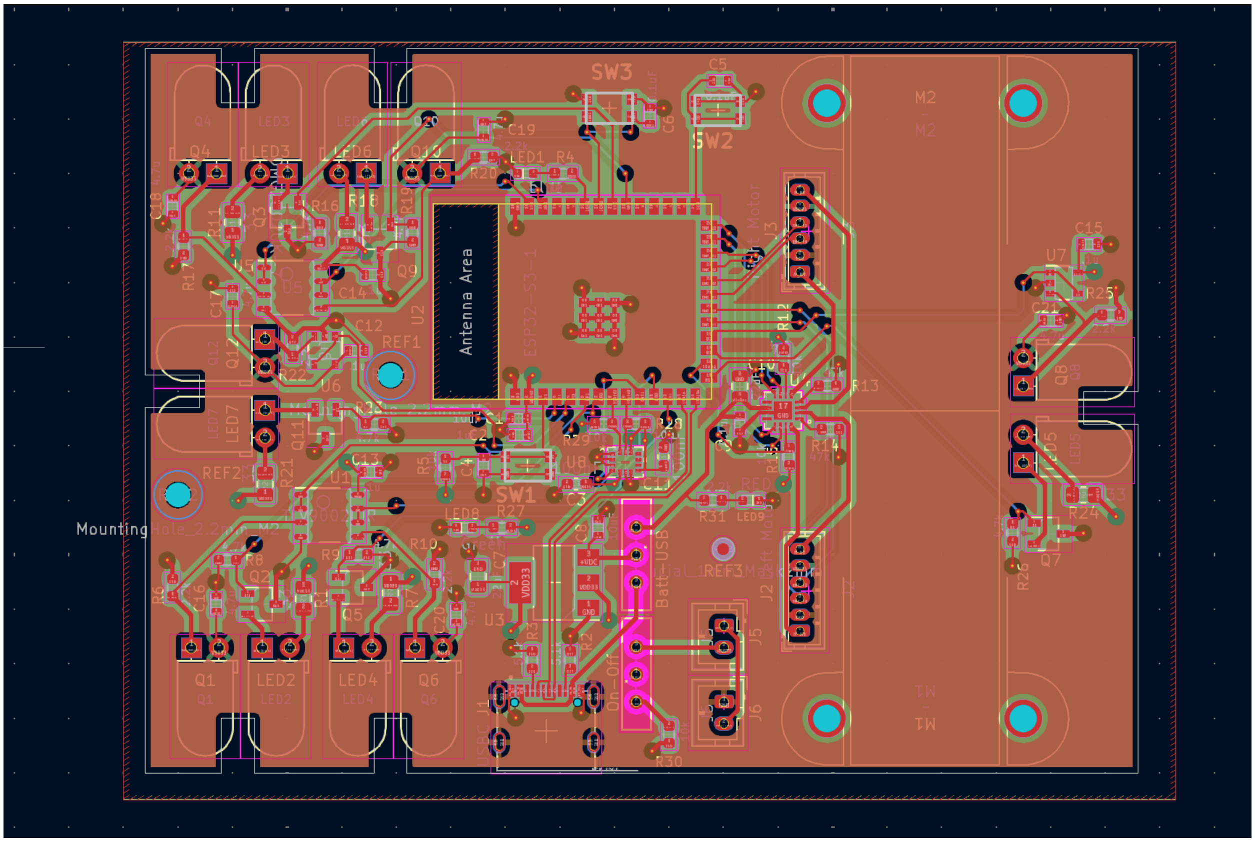
Appendix



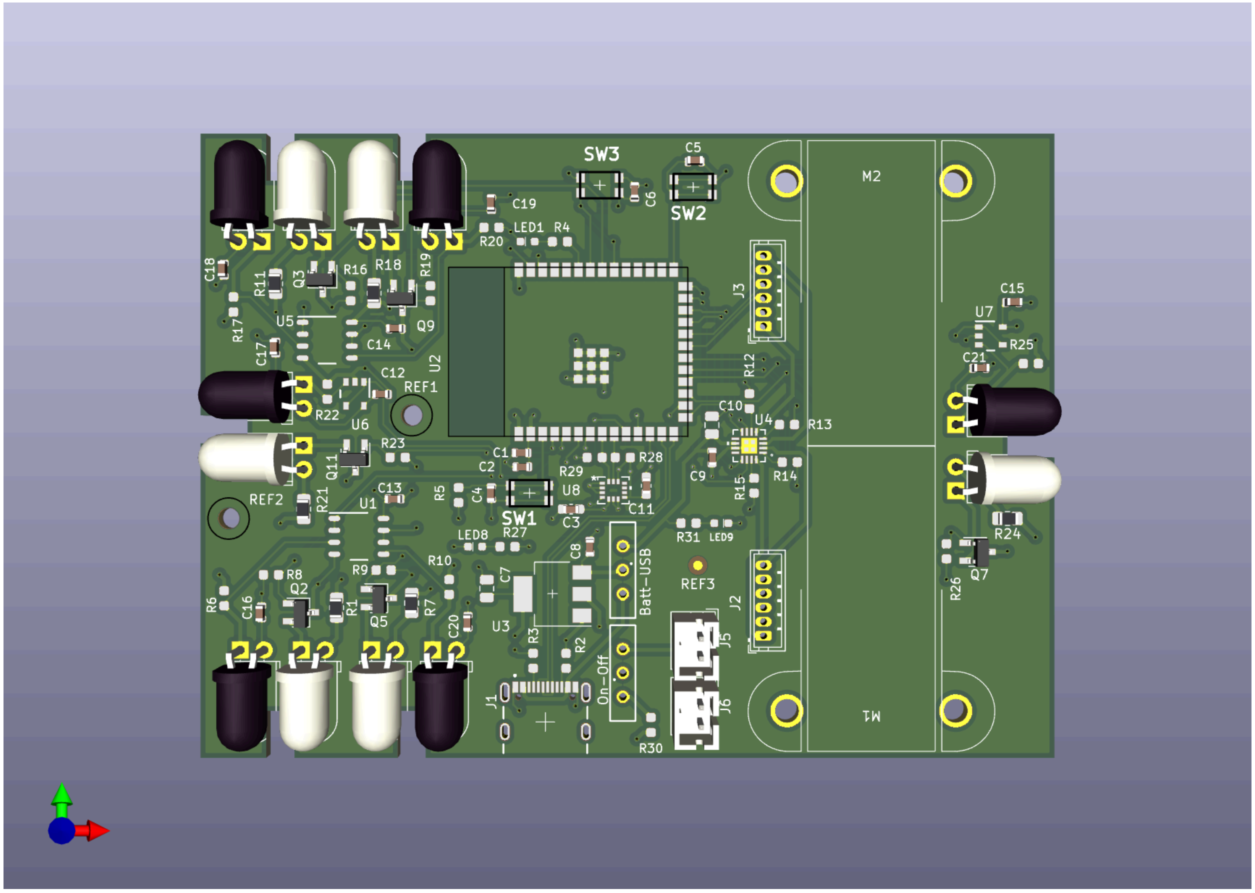
Mouse Board KiCad Schematic



Mouse Sensors KiCad Schematic



Mouse Board Layout



3D View of Mouse Board Layout