The NI-hicle (Navigation Independence)

James Beckett, Alexander Jenkel, Juan Velasquez

Group 20 – The Knights of NI

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

***Abstract*— This project implements navigational capabilities in a remote-controlled vehicle using various onboard sensors such as radar, ultrasonic, and an onboard 3D camera connected to a Graphics Processing Unit (GPU) and a Central Processing Unit (CPU). The various sensors feed their data to a central computing unit which will make the necessary decisions to either adjust the steering, adjust speed performances to accelerate or decelerate, avoid collisions or avoid obstacles and stop the vehicle.**

Keywords—autonomous vehicle, autonomy, GPU, 3-D camera, ultrasonic, radar

I. Introduction

The outcome of this project was to have a vehicle that can autonomously traverse a course that could feature turns of various angles, twists, hazards, obstacles, uneven surfaces and even dead ends. The vehicle would be capable to navigate through the course as quickly as possible without colliding and avoiding obstacles all while adhering to the safety and road standards. Additional features of the course could include reconfigurability of the track, where the walls and obstacles could be translated to create a new track, and the vehicle would still be able to navigate through it accurately.

The project is featured on a 1/10 scale model Remote Control Car (RC). Safety, cost, and time constraints restricted the project to be done on a smaller scale. Translating the algorithm, hardware, and technology into a "real-world" problem looks as simple as having a processor that responds in real time to different flags and interrupts in a system while still operating efficiently and correctly. This work could easily be applied to existing technologies such as Roomba or other autonomous vehicles that are already in production.

II. Design Overview

A. Initial Design

The initial design for our autonomous vehicle comprised a series of distinct sensors which fed distance and proximity data. The basic operational summary of the system is as follows. The 3-D stereo camera, being the primary sensor, will utilize its stereoscopic vision system to detect and determine depth of images in order to measure distance from detected objects at longer ranges and to facilitate course navigation. The radar module will send proximity target data to the Arduino Atmega 2560 in order to detect mid-range distance objects that are in the path of the autonomous vehicle. Ultrasonic proximity sensors will send data to the Arduino Atmega 2560 in order to facilitate minimal range object detection. This data will be fed to the Arduino Atmega 2560 in order to determine whether a motor or steering signal is required each cycle.

This project is sponsored by Dr. Guo, CECS, University of Central Florida

B. Current Design

Due to the COVID-19 pandemic, our current design is very similar to our original design. Our V1.0 PCB's were functional, so the urgent need for and inability to acquire a V2.0 PCB was mitigated, and the team was able to produce a partially functional system. However, some changes were able to be implemented from our initial design to our current design and are highlighted below.

One change implemented was that it was determined that the radar that we had initially selected and purchased was unable to provide the data that we needed in order to facilitate proximity readings. The API provided by the company simply was not enough for us to install and implement the device into our project, and so we instead opted to utilize a radar that generated an interrupt signal when an object was detected within a certain hardware-defined threshold. This interrupt was utilized to initiate the audible safety board, which would then play an installed *.wav* file through the system's speakers.

A rotary encoder was also initially going to be utilized to provide speed data to the Arduino Atmega 2560. It was determined that this component was unneeded as the system was not necessarily in need of speed data. While future revisions of the project may necessitate the use of speed data, this revision simply did not.

Another implemented change was the utilization of a lithium-polymer battery with a larger voltage. This change was made to power the Jetson Tx2's carrier board and accompanying peripheral devices. The power system PCB was initially designed for input voltages from 4.6 to 12VDC, so no modifications were required for the power system PCB to accommodate the larger input voltage.

III. Goals and Objectives

The goals and objectives for this project are:

- To produce an autonomous vehicle with the capability to navigate a reconfigurable course without striking the course walls or another vehicle.
- To provide designs and techniques that can be utilized by our sponsor in his research of autonomous vehicles
- To race our design in a competition

IV. Requirements

The project's requirements and specifications are presented in Table 1.

TABLE 1
PROJECT REQUIREMENTS AND SPECIFICATIONS

Max Height	1	Foot
Max Weight	15	Pounds
Object Size Detection	6x12	Inch
Object Detection Range	1	Meter
Autonomy [1]	4/3	SAE Level
Object Detection Response Time	1	Second
Object Detection Response Time	1	second
Stopping Distance from 5mph	2	ft
Object Response Distance	3	ft
Minimum Distance from Obstacle	6	in
Max speed	10	mph
Acceleration time from Full Stop	10	seconds
Stop Time from Max Speed	5	seconds
Source Voltage	11.1	V
Source Capacity	5000	mAh
Down Converted Voltages	1.8, 3.3, 5	V
Battery Type	Rechargeable	--
User Adjustment to Autonomy Level	0/1	SAE Level

V. Research and Theory

Autonomous Vehicles are a thriving emerging technology that already has had great impact, which may revolutionize transportation, while substantially enhancing traffic safety and efficiency. Market interest for autonomous vehicles currently is for the purpose of delivering goods or ride sharing with a long-term goal of reaching Level 4 or Level 5 [1] autonomy within 10 years. Since its inception, the idea to create autonomous vehicles is ongoing, avid, and ambitious. Remote controlled car competitions are popular projects that dwell on this idea of full automation, which allows to develop, work and create new technology.

A. *GPU Image Processing*

The Graphics Processing Unit (GPU) is not only a powerful graphics engine but also a highly parallel programmable processor featuring peak arithmetic and memory bandwidth that substantially exceeds a dedicated Central Processing Unit (CPU). Graphics Processing Units can be utilized to apply texturing and pixel engines that were originally designed for 3-dimensional modeling and rendering, to many classic image-processing problems to provide speed increases over CPU-only implementations, without comprising image quality.

Compute Unified Device Architecture (CUDA) is a general architecture for parallel computing introduced by NVIDIA in November 2007. It includes a new programming model, architecture and instruction set oriented towards parallel computing. This allows pixels to be treated in parallel. In the CUDA programming framework, the GPU is viewed as a compute device that is a coprocessor to the CPU.

The GPU has its own DRAM, referred to as device memory, and executes a very high number of threads in parallel. More precisely, data-parallel portions of an application are executed on the device as kernels which run in parallel on many threads. In order to organize threads running in parallel on the GPU, CUDA organizes them into logical blocks. Each block is mapped onto a multiprocessor in the GPU. All the threads in one block can be synchronized together and communicate with each other. Because there is a limited number of threads that a block can contain, these blocks are further organized into grids allowing for a larger number of threads to run concurrently. CUDA also supports the use of memory pointers, which enables random memory-read and write-access ability. In addition, the CUDA framework provides a controllable memory hierarchy which allows the program to access the cache (shared memory) between GPU processing cores and GPU global memory.

B. *Stereo Vision Systems*

Human beings acquire information about the location and other properties of objects within an environment thanks to a powerful and sophisticated vision system. The perception of a third dimension (depth) occurs due to the difference between images formed in the retinas of the left and right eyes. In the process of image formation, the catches of each eye are not equal because they present a slight variation in the position of the observed objects, attributed to the separation between the eyes. Artificial stereo vision systems are generally inspired by the biological process to extract three-dimensional information from digital images, which can be used to perform 3D reconstructions, tracking, and detection of objects.

There are several devices that provide three-dimensional information, depending on the operating technology they can be classified into stereo vision sensors, structured light devices or sensors based on the principle of Time of Flight (ToF). These devices are used in several areas with multiple purposes, in Robotics they are employed as essential tools in navigation applications, three-dimensional parts review, among others.

However, depth data provided by stereo devices have errors attributed to several aspects related to cameras hardware and computational processes that are performed to obtain these values. It is possible to enumerate some sources of errors as hardware system error, camera calibration error, feature extraction and stereo matching errors. These inherent errors that such data present should be considered in the applications where depth data generated by 3D vision sensors are used, such an example is the Robotic Vision. In real applications, such as autonomous robotics, it is important to consider and treat those visual errors in order to achieve correct decision-making process during a navigation task, for example.

As said, humans can have a three-dimensional perception of the world through the eyes due to the difference observed in the images formed in left and right retinas. In the imaging process, the images sent to the brain from each eye are not the same, with a slight difference in the position of the objects due to the separation between the eyes, which form a triangle with the scene points. Thanks to this difference, by triangulation the brain can determine the distance (depth) that the objects are in relation to the observer position. The implementation of stereo vision in computers uses this basic principle to recreate a 3D scene representation based on the two images of it taken from different viewing points. This is known as stereo reconstruction. In order to do stereo reconstruction, a series of steps are necessary, as calibration, rectification, and further depth determination.

The calibration process estimates intrinsic and extrinsic parameters of the cameras. Intrinsic values include the focal length, principal point coordinates, radial and tangential distortion factors. They are commonly used to obtain images without distortions, caused by the lenses and camera construction process, and to obtain three-dimensional representations of a scene. On the other hand, extrinsic parameters relate the real-world reference systems and the camera, describing position and orientation of the device in the real-world coordinate system (i.e. rotation matrix and translation vector). In addition to the calibration (for each camera), may be developed a stereo calibration, this process allows obtaining information that relates the positions of the two cameras in space.

Stereo rectification is the process in which a pair of stereo images are corrected, so that, it appears that they had been taken by two cameras with row-aligned image planes as shown in Figure 2. With such process the principal rays of the cameras are parallel, that is, they intersect at infinity. This step facilitates the stereo disparity estimation, a fundamental process prior to the estimation of the depth map.

The stereo camera computes depth information using triangulation (re-projection) from the geometric model of non-distorted rectified cameras. Assuming the two cameras are co-planar with parallel optical axes and same focal length, the depth of each point is calculated. In this calculation, depth varies inversely proportional to the disparity between baseline distance and image distance.

C. Radar Sensors

One of the leading technologies that is being used in the automobile industry is the radar-based safety system. Radar is being used for blind spot detection, automatic emergency braking, pedestrian automatic emergency braking and forward collision. Cameras and radar are now being used in the Advanced Driver Assistance Systems (ADAS) to provide lane-departure warnings and adaptive cruise control that allows the vehicle to follow the vehicle in front. As more ADAS systems become more advanced, they are expected to become government-mandated in the future following the recent introduction of legislation such as rearview video systems in vehicles and advanced emergency brake assist (AEB) for commercial vehicles.

Many automotive radar systems use a pulse-Doppler approach, where the transmitter operates for a short period, known as the pulse repetition interval, then the system switches to receive mode until the next transmit pulse. As the radar returns, the reflections are processed coherently to extract range and relative motion of detected objects.

D. Ultrasonic Proximity Sensing

Ultrasonic proximity sensors are a common type of proximity sensor that works by emitting sound frequencies higher than the audible range of human hearing. The basic principle behind this type of sensor is that the sensor emits an ultrasonic pulse and receives it back. The time difference between transmission and reception is used to determine the distance traveled. Since the ultrasonic pulse will bounce off of an object, the distance travelled will indicate the distance to the object. Since ultrasonic proximity sensors utilize sound instead of light, they can be used where photoelectric sensors have difficulty, such as in strong sunlight. This type of sensor is also immune to common contaminants such as dust and moisture. This type of sensor would be susceptible to noise interference from any similar devices emitting pulses with the same sound frequency and potentially provide false readings to the microcontroller. This may be detrimental during a competition where multiple vehicles may be operating with similar sensors and their frequencies emissions may interact unfavorably.

E. Servo Motors

Servo motors have been around for a long time and are utilized in many applications. They are small in size but pack a big punch and are very energy efficient. These features allow them to be used to operate remote-controlled or radio-controlled toy cars, robots and airplanes. Servo motors are also used in industrial applications, robotics, in-line manufacturing, pharmaceutics and food services.

To fully understand how the servo works, you need to take a look under the hood. Inside there is a pretty simple set-up: a small DC motor, potentiometer, and a control circuit. The motor is attached by gears to the control wheel. As the motor rotates, the potentiometer's resistance changes, so the control circuit can precisely regulate how much movement there is and in which direction. When the shaft of the motor is at the desired position, power supplied to the motor is stopped. If not, the motor is turned in the appropriate direction. The desired position is sent via electrical pulses through the signal wire. The motor's speed is proportional to the difference between its actual position and desired position. So, if the motor is near the desired position, it will turn slowly, otherwise it will turn fast. This is called proportional control. This means the motor will only run as hard as necessary to accomplish the task at hand.

Servos are controlled by sending an electrical pulse of variable width, or pulse width modulation (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movement. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in both the clockwise or counter-clockwise direction. The PWM sent to the motor determines position of the shaft and based on the duration of the pulse sent via the control wire; the rotor will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position. Shorter than 1.5ms moves it in the counterclockwise direction toward the 0° position, and any longer than 1.5ms will turn the servo in a clockwise direction toward the 180° position. The PWM signal effect on a servo motor is depicted in Figure 8.

When these servos are commanded to move, they will move to the position and hold that position. If an external force pushes against the servo while the servo is holding a position, the servo will resist from moving out of that position. The maximum amount of force the servo can exert is called the torque rating of the servo. Servos will not hold their position forever though; the position pulse must be repeated to instruct the servo to stay in position.

VI. HARDWARE DESIGN

Strategic component and parts selection were conducted in order to fulfill engineering specifications, requirements and standards. Selected components are broken down into three subsections: platform, payload and peripherals. The platform consists of the vehicle chassis and the components required to power, propel and steer the vehicle. The payload consists of the electronic components required to process all peripheral data and provide motor and steering control signals to the platform. The peripherals consist of all secondary sensors utilized to facilitate course navigation and collision avoidance.

A. *System Overview*

The basic operational overview of the system is as follows. The 3-D stereo camera, being the primary sensor, utilizes its stereoscopic vision system to detect and determine depth of images in order to measure distance from detected objects at longer ranges and to facilitate course navigation. The radar module sends proximity target data to the Arduino Atmega 2560 in order to detect mid-range distance objects that are in the path of the autonomous vehicle. Ultrasonic proximity sensors send data to the Arduino Atmega 2560 in order to facilitate minimal range object detection. This data is fed to the Arduino Atmega 2560 in order to determine whether a motor or steering signal is required each cycle.

A system block diagram is included to summarize and provide a functional overview of the project. This diagram is presented below in Figure 1.

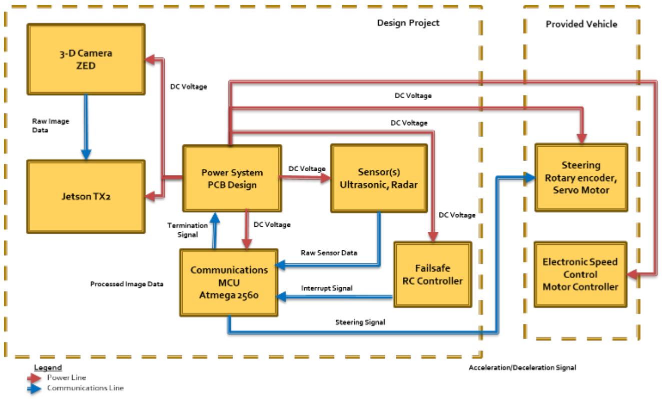

Fig. 1 – System Block Diagram

Input power will be supplied by a rechargeable Lithium Polymer battery supplying 11.1VDC at 5000mAh. This voltage and current will be routed to the power systems PCB where the 11.1V will be directed to three linear voltage regulator circuits that will convert the 11.1V input to 1.8V, 3.3V and 5V respectively. The 11.1V will also be routed through the power system PCB as an unregulated input that will be directed to the motor controller to provide voltage and current to the drive motor and to the image processor. The power system PCB will output 1.8V, 3.3V and 5V to the main payload PCB which contains the Arduino Atmega 2560 and safety circuits. The Arduino Atmega 2560 will utilize the 3.3V to power itself. The 1.8V, 3.3V and 5V voltages will also pass through the main payload PCB to be directed to the peripherals sub-systems.1.8V will be routed to the radar module. 5V will be directed to and utilized by the ultrasonic proximity sensors and 3-D stereo camera. System wide power flow is illustrated below in Figure 2.

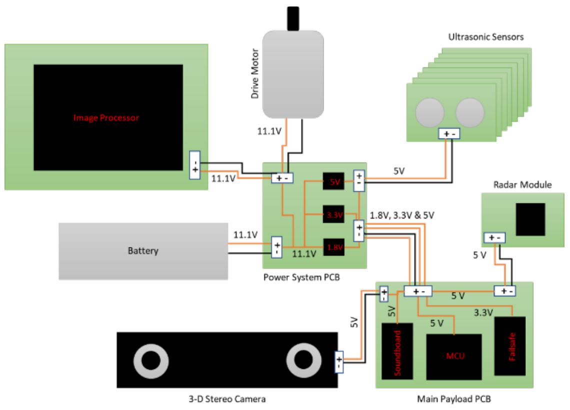

Fig. 2 – Voltage Diagram

Raw image data will be transferred from the 3-D stereo camera to the image processor. Rectified image data will then be passed to the Arduino Atmega 2560. The MUC will also accept proximity data from the ultrasonic proximity sensors and the radar module to facilitate object detection. The MUC will then determine motor and steering output signals to facilitate collision avoidance. Signal flow throughout the system is depicted below in Figure 3.

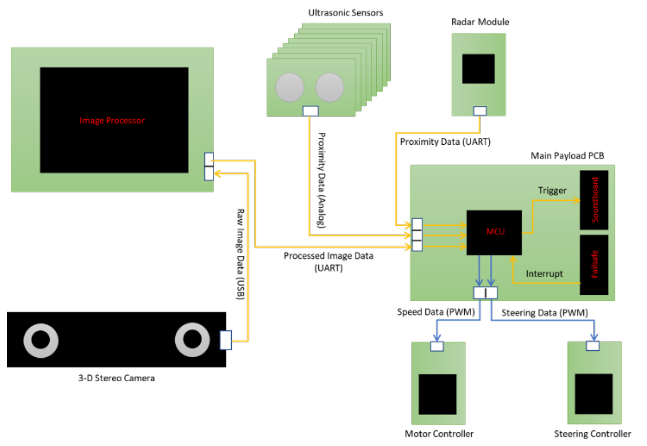

Fig. 3 – Signal Flow Diagram

B. *Platform*

The sponsor-provided vehicle chassis is the Traxxas Ford Fiesta® ST Rally Radio Controlled 1/10th scale car. This specific vehicle chassis would meet our intended goals for autonomous vehicle as well as confirm our research into vehicle platforms. The hybrid of on-road and off-road capability as well as four-wheel drivetrain allows the autonomous vehicle to function over a broader range of terrains.

The vehicle chassis was modified to carry the sensors and payload. A dual-level plexiglass mounting service was fabricated to mount the PCB's, Sound board and Jetson Carrier board. We utilized a dual-level design to enable the system's custom wiring harnesses to be routed between the layers to prevent accidental unplugging of the harnesses in case the vehicle got too close to an object.

The power management PCB provides regulated DC power to all subsystems. The power system's input is a rechargeable lithium-polymer (LiPo) battery pack operating between 4.6 and 12 VDC, providing current between 3500 and 7500 mAh. The power management PCB regulates the input voltage via DC-DC buck voltage regulators which provides the required voltages for the various components of our system at 1.8V, 3.3V and 5V. Each regulated voltage is provided by its own voltage regulator circuit to enable quick corrective measures without replacing the entire PCB.

The drive motor that was supplied with the vehicle was the Traxxas Titan 12T 550. The group opted to replace the provided brushed motor in favor of a brushless motor for racing applications. When switching to a brushless motor, the system also required changing the provided electronic speed controller to one which could provide the required signals to the new motor. The electronic speed controller that the group utilized was the Traxxas VXL-3S. The steering servos provided with the vehicle were considered acceptable for our application and were unchanged.

C. *Payload*

The Arduino Atmega 2560 is the brain of the design and it takes the inputs from the sensors, interprets the data and provides output commands to the vehicle to navigate and avoid collisions. The Arduino Atmega 2560 is constantly taking in data that the GPU feeds it and adjusts the motor, speed controller, and steering servos accordingly. Additionally, the Arduino Atmega 2560 monitors any wireless communications and awaits a manual override signal—which will be supplied by the user (if necessary) as a failsafe technique—and then "listens" to the user's instructions in lieu of making its own. The Arduino Atmega 2560 chosen for our design was the Arduino ATMEGA2560-16AU. This processor was chosen due to its number of I/O pins, communications protocols, UART channels and the wealth of Arduino libraries available.

The image processor we chose is the Jetson TX2, which takes raw data from the 3-D camera and provides data to be processed by the Arduino Atmega 2560 to avoid collisions. The ZED Stereo Camera has an impressively large amount of data that it supplies in every frame, but for our intents and purposes all we will use is the distance measurement features. The ZED Stereo Camera passes a 1280 x 720 point cloud to the Jetson Tx2 where every single point in the grid contains the measurement for that pixel in the image that the camera captures. While we may not necessarily need the whole point cloud to make our project, this will be more than sufficient for navigational capabilities.

Due to the minimal sound generated by an electric motor and small vehicle platform, an audible safety device was selected in order to alert nearby pedestrians of the presence of the autonomous vehicle. The Adafruit Audio FX Mini Sound Board is an efficient, cost effective means to alert nearby pedestrians of the presence of the autonomous vehicle and is configurable with up to 2MB of storage for various audible alerts recorded in compressed or uncompressed MP3 or WAV format. Our team decided to implement movie sound bites from the motion picture "Monty Python and the Holy Grail".

An electronic failsafe was to be designed to mitigate liability associated with the operation of an autonomous vehicle. Unfortunately, due to the COVID-19 pandemic and required social distancing, the failsafe was unable to be fully developed. We had the pinout available on the Arduino Atmega 2560 to receive a level-change interrupt, but without being able to test the receiver with an oscilloscope it was really impossible to properly test and develop the interrupt signal. The failsafe would have functioned in two separate ways. The first operation would have been to act as a user-controlled override of the steering and speed functions of the vehicle. The existing remote-control functions provided with the initial vehicle would have been integrated into our design, thus allowing an operator to seamlessly take control of the vehicle to avoid injury to individuals or damage to property. This option brings the autonomy level from a 4 down to a level 0 ([1]). The second operation of the failsafe would have been as an electronic "kill switch" that immediately disconnects power to the motor, thereby disabling any powered vehicle movement, but still allowing the vehicle to process and steer away from obstacles. This operation would have been important in case the vehicle travelled outside the range of the existing remote-control functionality present in the original vehicle. This option would alter the autonomy level from a 4 to a level 3).

D. Peripherals

A stereoscopic camera was provided by our sponsor, Dr. Guo. The sponsor-provided camera was chosen for this project as a proof of concept for optical image directed, obstacle avoidance. The sponsor-provided camera is the ZED Stereo camera and is a 3-D sensor which contains depth perception and motion tracking functionality. The ZED device is composed of stereo 2K cameras with dual 4MP RGB sensors. It has a field of view of 110° and can streams uncompressed video at a rate up to 100 FPS in WVGA format. It is an UVC-compliant USB 3.0 camera backward compatible with USB 2.0. Left and right video frames are synchronized and streamed as a single uncompressed video frame in the side-by-side format. Several configurations parameters of on-board ISP (Image Signal Processor) as resolution, brightness, contrast, saturation can be adjusted through the SDK that is provided by ZED development team. This camera has a compact structure and reduced size, compared to other stereo cameras. These characteristics make it relatively simple to incorporate into robotic systems or drones.

To aid in the aspects of collision avoidance, radar sensors will be implemented because of the range span that the sensors provide. Radar can propagate at high frequencies and are able to detect objects within millimeters of the vehicle. The SEN0192 Motion Detector can detect movements in a room, yard, or even on the other side of a wall. It's a Doppler radar sensor that operates in the X-band frequency at 10.525 GHz and indicates movements with oscillations in its high/low output. Sensitivity is manually adjustable with a potentiometer on the back of the device, offering direct line of sight detection from roughly 8 to slightly over 30 ft (~2.4 to 9+ m).

Ultrasonic proximity sensors are assumed to be the most reliable of the secondary proximity sensors to be utilized in the project. Their low cost, effective range and speed are suitable for use as backup collision avoidance sensors. Ease of programming and integrating is also a factor when selecting this type of sensor for its intended purpose. The HC-SR04 sensor is ranging module that provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules include ultrasonic transmitters, receiver and control circuit. The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back. If the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning. Test distance = (high level time x velocity of sound (340M/S) / 2.

VII. Software design

The software of the autonomous vehicle will be that of an embedded system, hence techniques such as reusability and portability will not be considered—since the UCF1/10 team uses a nearly identical set up. Instead the main focus of the software will be that of correctness, reusability, reliability, and efficiency. The project software behavior will be similar to that of a Roomba device, where the end user will only power the device, and then the vehicle will begin to navigate through a track, avoiding obstacles along the way. Since the project will feature different sensors, controllers and multiple processors, UART, SPI and Serial communications will be used throughout the project with as few devices on each protocol as possible. The ultrasonic and radar sensors are analog devices, and as such either timed pulses or level changes—i.e. interrupt triggers. It is essential to have a high baud rate between the sensors and the Arduino Atmega 2560 because the vehicle will be in constant motion and will need to update its positional data as quickly as possible, which in turn will help process the data that is being collected by the sensors.

The GPU will interpret the data and calculate the distances for any objects that are potentially spotted by each sensor. Once the data is collected the Arduino Atmega 2560 will perform calculations and it will determine if either a threshold is triggered—signifying that evasive actions are necessary—or if the vehicle will be able to operate normally on the same path. The GPU will receive a point cloud from the ZED Stereo Camera where every point in the aforementioned cloud has distance data for the respective pixels. It is the job of the Jetson Tx2 to take this point cloud and make it legible to the Arduino on the Communications PCB. There was great deliberation about the method of interpretation for the point cloud data, where the main two selections are algorithmic approaches or utilizing Machine Learning procedures. The latter was favored at the start of the project, as the ZED Stereo Camera features API specifically for machine learning algorithms that contain datasets for detecting a multitude of everyday objects. After much contemplation, it was deemed that machine learning was both too robust and too powerful for what the project actually needed. While an algorithmic approach may be too simplified, our project does not actually need to detect the type of object that is present, it only needs to detect the presence of an object, which can be done by reading the point cloud data and applying basic statistical analysis to the cloud. In our case, the basic statistical analysis is just averaging over specific ranges and then comparing said range-averages. While this may not yield exact results for where objects are, or where they are not, the goal of the ZED Camera is merely to give the Arduino Atmega 2560 an idea of where an object *could* be, while the actual object avoidance and navigation will be achieved via the onboard sensors.

Serial communication will be used to communicate with the ZED Stereo camera. This communication protocol is used because the Jetson TX2 has an onboard USB port and it will lessen the size of the software. The ZED Stereo Camera is also compatible with ROS and has well-documented Python and C++ API, which will cut down on the amount of programming that we will have to do. The only parts of the ZED Stereo camera that we will need to program will be the minimum required to interface the ZED Stereo camera to the Jetson TX2.

The software will feature different Interrupt Service Routines (ISR) to handle the different evasive maneuvers the vehicle will exhibit. Some of the evasive maneuvers the vehicle could make are stop, produce a warning sound, steer either left or right to various degrees, accelerate and decelerate. Setting up the ISRs as functions will ease debugging the software as the various behavior-controllers will be centrally located within the ISR. This also allows more than one person to work on or to troubleshoot the code if need be.

Stopping the vehicle will need to happen once the vehicle gets to within a certain threshold distance away from an obstacle. The sensors will send the objects' distance data to the

Arduino Atmega 2560 which will then determine which action to take. According to the requirements from Table 1, the vehicle will enter the ISR for stop/decelerate/accelerate actions when the sensors detect and object that is three feet away. At three feet the vehicle will commence deceleration while the sensors gather additional data from the surroundings in case the vehicle will also need to turn to avoid the obstacle. If the object does not go below three feet, the Arduino Atmega 2560 will send a command to the motor controller to accelerate, but if the object goes below two feet the ISR will break the connection to the motor controller to make the vehicle come to a complete stop. In addition, the software will feature a failsafe ISR that is intended to be a "kill switch" for the vehicle.

A kill switch is required for the competition and is specified by competition standards where the user will have the ability to stop the vehicle by flipping a switch or pressing a button. The ISR will be triggered by an output signal from the user's remote control that is compatible with the OEM radio antenna. The end user should call the kill switch when the user sees a flashing LED mounted on the vehicle—indicating some form of onboard error that is directly impacting the vehicle's navigation, or in addition the LED can also alert that a crash is imminent and unavoidable—or if the user hears the warning sound play from the onboard speakers. This LED will be triggered by the sensor data when the vehicle reaches the different thresholds. At two feet the LED will begin to flash slowly to let the user know that the vehicle should begin to stop. At six inches the LED—or possibly multiple LEDs—will flash rapidly to let the end user know that the vehicle should come to a complete stop within the required time of five seconds. If the vehicle stops then the end user will disregard the waring LED. If the vehicle appears to continue going, then the end user will simply pull the switch trigger from the remote control.

To steer the vehicle in the correct speed and direction, measurements will be taken from the steering controller unit. Based on distance data received from the various sensors, speed read from the rotary encoder (or possibly calculated from changing distance data), and radius of the curve measured from distance data, the vehicle will be able to run an algorithm that correctly decides on which direction or speed to take. One of the main components that will be featured in the navigational ISR is the 3D camera. The camera has the ability to sense object distance and can create a point cloud. One of the abilities given by the Jetson Tx2 is the ability to pass the calculations off to the GPUs and accelerators that are included in the architecture. This will allow the computations to be done very quickly, which in turn allows us to have more computations in a given session. Software will be made to parse through the point cloud and generate a "collection" of objects along with their distances and size. Once both size (needed for steering direction) and distance (needed for reaction time) are found, the Jetson will send the data to the Arduino Atmega 2560. This will trigger the software to enter the ISR. The auxiliary sensors will also contribute data to have a more accurate calculation of the distances and will effectively act as a handshake with the ZED Stereo Camera. Once the path is determined, the Arduino Atmega 2560 will send a command to the steering control to turn to the appropriate angle.

As a safety feature it was decided to include an audible device to make the vehicle stand out in the event that there are bystanders that are unaware of the oncoming vehicle. Different sounds will be programmed when the vehicle performs an evasive maneuver, or to indicate different operation statuses. In regard to evasive maneuvers, the vehicle will produce a warning sound when approaching an obstacle, when it has come to a complete stop, when the vehicle goes in reverse, as well as if any sensor or car component fails. In addition, the car will play sound bytes once the program has been initialized and had no errors, and again once the program exits successfully. This functionality was decided to be separate from the stop/accelerate/decelerate function to reduce the size of the function and ease of debugging, but the Arduino Atmega 2560 will still be able to control the speed separately from the ISR.

A software class diagram for the system is presented below in Figure 4.

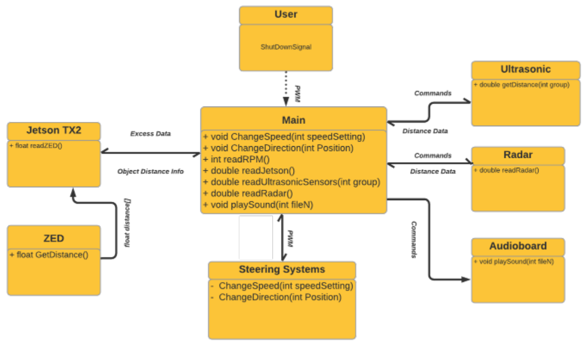

Fig. 4 – Software Class Diagram

The software class diagram can be broken down into three distinct sections: User interrupt, peripheral operations, and steering systems. The User will always have the ability to send an interrupt through the OEM remote control, which will disengage the motor while also allowing the car to navigate. The Arduino Atmega 2560 will also be reading and reacting to distance data received either from the radar, ultrasonic sensors, or Jetson Tx2. Depending on the scenario, the Arduino Atmega 2560 will have the decision to make with respect to speed, navigation, and also whether or not it should play a sound byte. Since this process needs to be fast—especially in a racing setting—the software should be kept to a minimal degree so that the vehicle can be as responsive as possible. Another facet of the software class diagram that should be considered is that there are only two interrupts in the system: the User's kill signal interrupt, and the radar. Keeping the number of interrupts to a minimum guarantees that the program will flow almost continuously.

Now that the classes have been sorted out, it follows that the next step is to determine the nature of the software and the order that the events should occur. This is done in Figure 5 below:

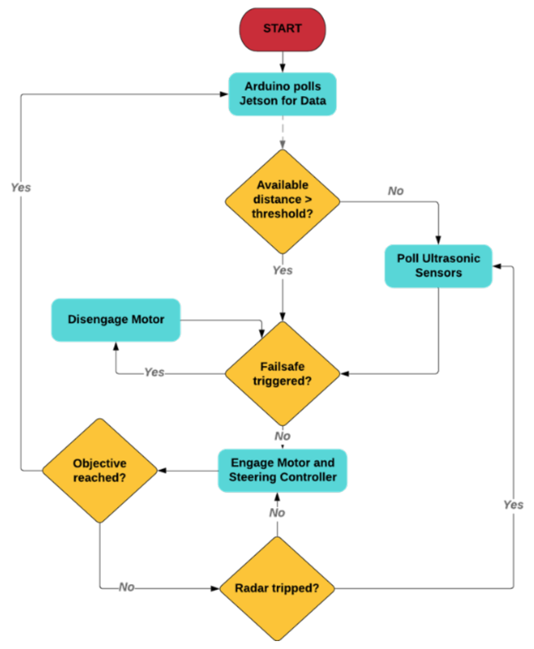

Fig. 5 - Software Flowchart

The software flowchart is essentially one big loop due to the nature of the Arduino family, which is to initialize and then to loop repeatedly until the programmer decides that it wants to terminate the program. Here is no different and the flow is sequential. The very first thing that will be done in every test run is that the Arduino Atmega 2560 will poll the Jetson Tx2 for any available distance data. The Arduino Atmega 2560 has a software-defined threshold value that it will test the distances against. If there is a direction that the NI-hicle can drive in that exists beyond the threshold, it will do so. In the case where either the distances are beyond the threshold, or if the radar is tripped while the car is in motion, the Arduino Atmega 2560 will begin polling the ultrasonic sensors. The Arduino Atmega 2560 will poll the sensors based on the fact that the ultrasonic sensors are better for reading measurements that are closer, whereas the ZED Stereo Camera has a minimum distance accuracy of 20cm. Utilizing the ultrasonic sensors will allow the Ni-hicle to travel in a more fine-tuned way than would be possible with only using the ZED Stereo Camera.

The final characteristic of the software flowchart is that before engaging the motors, the NI-hicle will check to see if the failsafe signal had been received. If the failsafe had been toggled, the program will wait until the user toggles the failsafe again. This will allow the user to take control of the NI-hicle should it make a wrong decision or go off track at any point.

VIII. Conclusion

Results have been promising, as we met many of our milestones for the project. The disruption caused by the COVID-19 epidemic has stalled progress on making the project fully functional, but the project overall is at 90% completion.

Without the interruption presented by the pandemic, the team feels that our project could have been fully functional and operated precisely as designed. Challenges for future teams that may continue our design work, which the team have identified, include integration of the ZED camera, integration of a “kill switch”, refining motor control capability and the lack of technical documents due to proprietary components.

Acknowledgment

The Knights of NI would like to thank the faculty, who have been instrumental in guiding our progress during this project, to include: Dr. Richie, Dr. Wei and Dr. Guo. The Knights of NI would also like to extend our gratitude to our panel for their attention, patience and understanding during these trying times and their outstanding ability to provide valuable feedback on our efforts.

References

[1] Matthew.lynberg.ctr@dot.gov. “Automated Vehicles for Safety.” *NHTSA*, 17 Sept. 2019, www.nhtsa.gov/technology-innovation/automated-vehicles-safety.