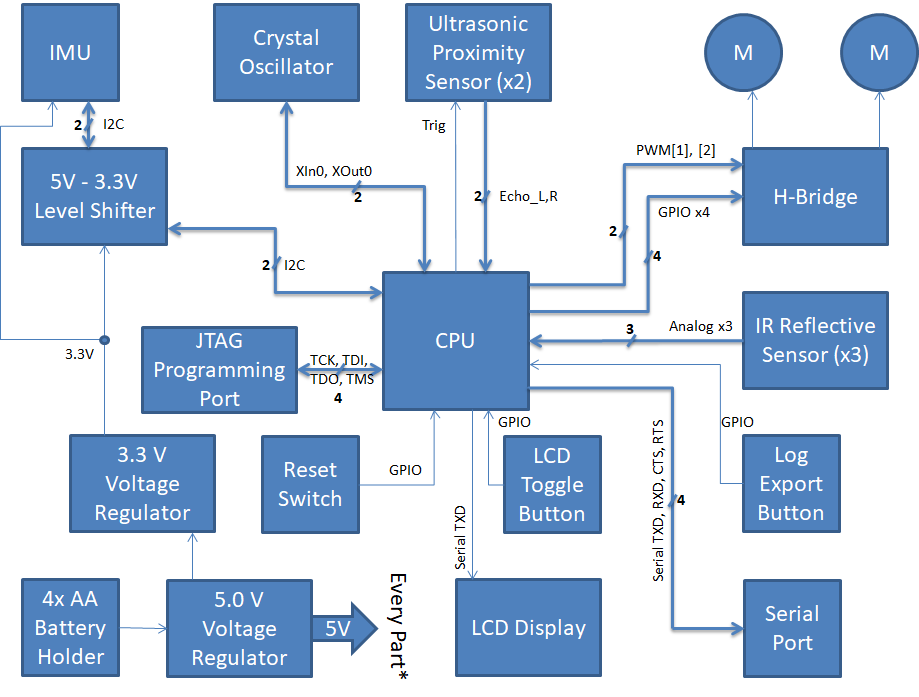
***Abstract-* This electronic document is a technical article that aims to describe the design and specifications of a computer-controlled car that can navigate a manually designated path. The car will monitor environmental feedback using Infrared reflective sensors, ultrasonic proximity sensors, and an Inertial Measurement Unit (IMU). Data collected by these devices will be used to control the motors for following the manually specified path, circumnavigating obstacles on said path, and collecting the car’s travel data.**

I. MOTIVATION

The Engineers have decided to build and construct a car that follows a custom path made from one-inch blue painter’s tape, while being able to avoid the obstacles that stand in the car’s way. Intrigued by the new autonomous technologies currently being developed by companies such as Tesla, our group was inspired to create our own version of a car on a much smaller scale. The car uses a chassis measuring 180x155x3mm that includes two wheels placed on either side of the circular base from the center, measuring 64x64x28mm. Included were two DC motors with little documentation. To follow the tape path, an IR sensor PCB will be attached to the bottom side of the chassis toward the front of the car. Data collection will also be a feature included on the car. Some of the data being tracked includes the car’s distance traveled, velocity, and acceleration. This data will be shown on a LCD character display situated on the chassis. Finally, the car will utilize two ultrasonic distance sensors attached to the car to assist with obstacle detection and avoidance.

II. TOP LEVEL BLOCK DIAGRAM



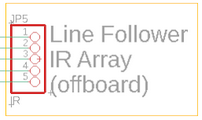
In total, the Autonomous car design consists of seven different sub-systems. Each component is listed and described below:

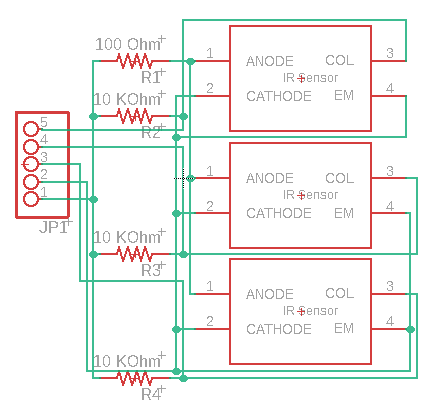
* *CPU:* AT32UC3C2512C-A2UT-ND, a 32-bit AVR microcontroller that has an operating voltage of 3.3V to 5V. With 64-pins, this microcontroller has more than enough functionality for this project.
* *H-Bridge and DC Motors:* L293DD combined with two DC motors allows the car to move.
* *IR Reflective Sensors:* QRE1113GR. Three of these sensors are to be soldered to a separate PCB that will be attached to the bottom of the chassis to implement line following functionality.
* *Ultrasonic Proximity Sensors:* Two HC-SR04 modules are placed toward the front of the car on either side with a 15 degree offset to prevent signal interference. The sensors will measure the distance of obstacles and help the car circumvent around them.
* *IMU:* SEN-13762, inertial measurement unit that is responsible for recording acceleration data of the car. From the acceleration, we can also find the velocity and distance travelled of the car.
* *Crystal Oscillator:* LFXTAL003210BULK. A 32.768kHz crystal oscillator that provides more accurate data capture through the IMU.
* *Level Shifter:* Consists of two BSS138’s. This allows the IMU to share its calculations in a way the CPU understands due to the differing voltage levels (3.3V vs 5V).
* *Battery Holder:* A wired 4-AA battery holder gives a voltage of 6V. This battery holder feeds into the 5V regulator.
* *Voltage Regulators:* Two voltage regulators are used in our design, a 5V and 3.3V regulator. The 5V regulator powers the CPU and the rest of the components. The 3.3V regulator is fed by the 5V regulator and powers the IMU.
* *LCD Display and Toggle Button:* Adafruit LCD Module receives data from the CPU and displays it as text on the screen. Toggle button is used to cycle through data (velocity, acceleration, etc).
* *Serial Port and Log Export Button:* RS232 Shifter. Allows for communication with a computer to export data collected by the IMU.
* *JTAG Programming Port:* 10-pin male pin header JTAG port used to connect a programmer to the CPU.
* *Power Switch:* Used to power and reset the board.

III. SCHEMATIC

The components previously listed make up the seven different subsystems. Each part is connected to the CPU. This section also goes into further detail about other components that are not associated with a subsystem. The complete schematic can be found at the end of this document.

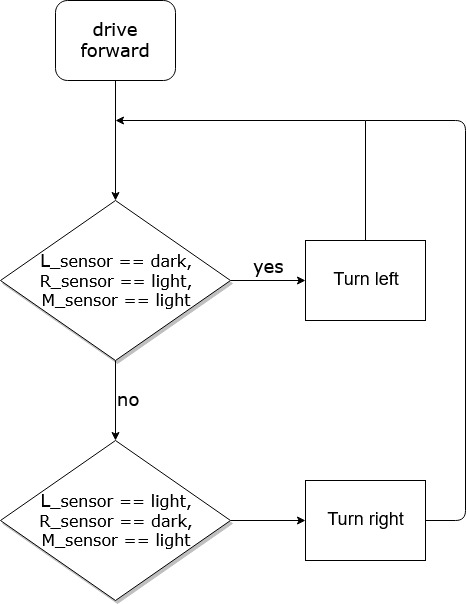
*A. Line Follower IR Array (offboard)*



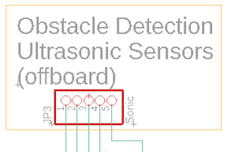


The IR Bumper board implements the line tape follower algorithm. The IR sensors on the board will observe reflective values of three spots in front of the car, one in the middle and one each to either side, and send readings to the CPU as an analog signal. The sensors will be placed on a breakout board that will be attached underneath the car towards the front bumper. A ribbon cable will then connect these sensors to a pin header on the PCB. If the CPU detects a change in readings from the sensor, it will execute one of two turning functions until the sensor output returns to the on-path values, then resume forward travel. A dark signal from the left sensor will encode for executing a left turn function, vice versa for right turn, and a dark signal in the middle with bright on the sides will encode for the on-path state. On the track there will be a wider section of blue tape that signals for the car to stop. In order for the car to stop, all three IR sensors must turn dark for a longer time. This section will extend for the length of three seconds. Once the IR sensors turn dark for three seconds, the CPU will read the signal and decide to stop. Pin 1 is the 5V VDD net and Pin 2 is the GND VSS net. Pins 3, 4, and 5 take the analog output of each of the IR sensors and feed them to the ADC inputs on the CPU.

Below is a flowchart to better understand the line tape follower algorithm for the IR bumper board:



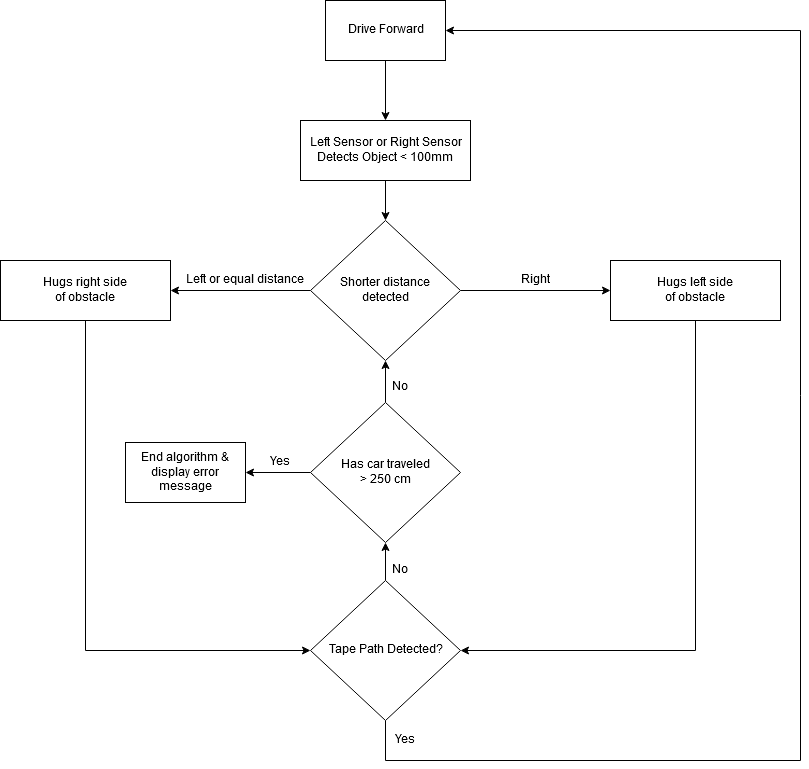
*B. Obstacle Detection/Circumnavigation (offboard)*



Obstacle detection will be implemented using two ultrasonic distance sensors to detect any objects in the way of the car. Each sensor will be placed with a 15 degree offset from the left and right sides of the car to avoid any ultrasonic interference from either sensor. If the CPU receives a reading from either sensor indicating an obstruction is within 100mm of the sensor, the car will activate the obstacle circumnavigation algorithm.

After deciding the initial direction to circumnavigate, the car will drive forward while “hugging” the obstacle. If circumnavigating on the right of the obstacle and the left distance sensor reads a drop in distance, the car will drive its right motor slower to arc the car away from the obstacle until the left distance sensor reading matches the hugging-distance of ~215 mm. If circumnavigating on the right of the obstacle and the left distance sensor reads an increase in distance, the car will drive its left motor slower to arc the car closer to the obstacle until the left distance sensor reading matches the hugging distance of ~215 mm. The circumnavigation on the left of the obstacle is vice versa.

If the car detects that it’s back on its original path, it will exit the circumnavigation subsystem and switch back to the path following subsystem. With obstacles fitting in a 60 cm x 60 cm footprint, the car should not need to travel more than 250 cm to circumnavigate an obstacle at worst, so if it does travel more than that while circumnavigating, the car will be programmed to stop and display an error message on the LCD display. From here, the car would have to be turned off and placed back on the tape path manually. Below is a flow chart to better understand the algorithm for the circumnavigation algorithm:

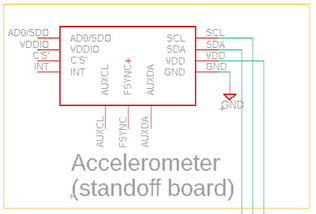


The distance sensors connect to the CPU using 4 pins: VCC, Trig, Echo, and GND. These sensors use a custom communication interface to send data to the CPU. Because of this, we will connect the two data pins to any available GPIO pins. The trigger pulse input pin (Trig) receives a pulse from the CPU, prompting the sensor’s transducer to send out eight 40 kHz signals. These signals will bounce on an object and back into the sensor. The echo pulse output pin (Echo) then receives this signal and performs a calculation to determine how far the object is from the sensor. The formula that the distance sensor uses for calculating the distance is the following:

Test distance = (high level time \* velocity of sound (340 m/s))/2

Pin 1 connects directly to the CPU to get a voltage of 5V. Pin 2 is connected to GND. Pin 3 serves as the Trig signal for both of the connected Ultrasonic sensors. Pins 4 and 5 handle the Echo signal of the right and left sensors respectively.

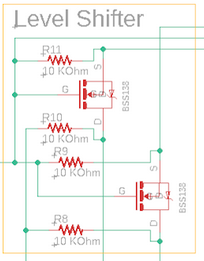
*C. Accelerometer*



Our accelerometer will come from an inertial measurement unit (IMU), the SparkFun IMU Breakout board. A 3.3V regulator supplies the needed voltage. Measurements from the accelerometer will be recorded every half second using the 32kHz crystal for accurate timekeeping. The IMU communicates accelerometer measurements across an I2C connection to the CPU. Due to the IMU operating on a logical voltage of 3.3V, a level shifter is needed for the 5V CPU to talk to the IMU. Although this module has more functionality, we will only be using it for the accelerometer, hence the disconnected pins.

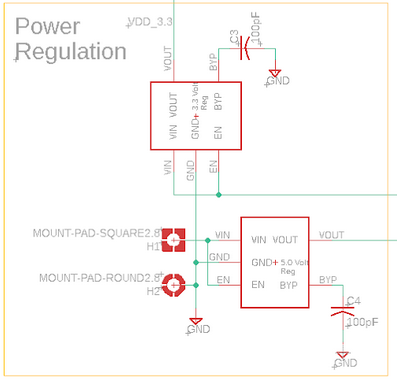
The accelerometer SCL and SDA signals are fed into N-channel MOSFETs acting as level shifters to translate between the 3.3V communications of the accelerometer and the 5V communications of the CPU.

*D. Level Shifter*



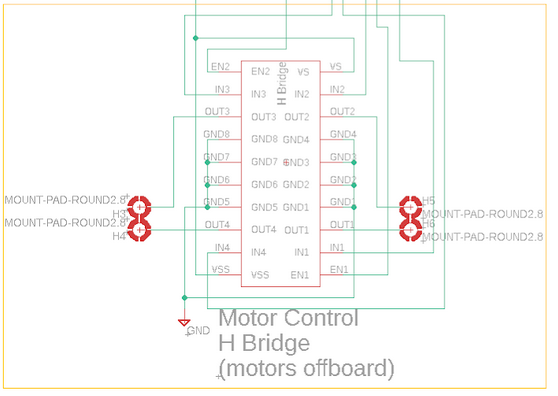
Because of the different voltages between the accelerometer and CPU (3.3V vs. 5V), we have included a level shifter circuit in our schematic. Our level shifter schematic was cloned from an off the shelf I2C level shifter module. By level shifting between the two components, the IMU will be able to share its calculations in a way that the CPU understands. The heart of the level shifter is a pair of N-channel transistors, the BSS138. Each transistor allows for one level-shifting channel. When paired with several resistors, the complete level shifting circuit allows for bi-directionality, being able to go from a low to high voltage or vice versa.

*E. Power Regulation*



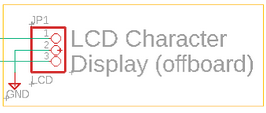
The 4-AA battery pack provides a voltage of 6V, which is incompatible with our board. To overcome this issue, the battery pack is fed into a voltage regulator that outputs 5V to the PCB. A switch on the car will power it on and off. The battery pack feeds into the VIN and EN pins of the regulator. GND pins are connected to ground as well. The BYP pin is connected to a 100pF capacitor, which is then connected to ground to reduce noise on the component. The capacitor setup was found on the component’s data sheet. VOUT is then connected to the respective power pins on the CPU. A separate 3.3V regulator is required to power the IMU. This regulator is connected in the same fashion as the 5V regulator with the exception of VOUT feeding into the VDD pin of the IMU through a series of resistors and level shifters found in the level shifter subsystem.

*F. Motor Control*



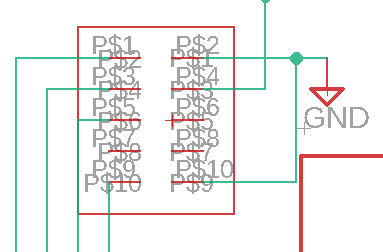
The motor driving subsystem will control the speed at which the motors run and the direction they move the vehicle in. Steering will be approached by running the left and right side motors at equal speeds in opposite directions to rotate the car in place, or running one motor faster than the other. To implement this, we will have the CPU send GPIO enable signals to the IN pins on the L293DD H-bridge IC to change the polarity of current through each motor. Another set of PWM connections from the CPU to the L293DD EN pins can be used to vary the average voltage supplied to the motors and produce variable speed control for the motors. The L293DD can control the polarity and speed of the two motors independently.

*G. LCD Character Display*



The LCD Character Display that we are using has a display format of 16x2 characters, allowing for up to 32 custom characters to display at one time. On the display we will display various things such as the car’s acceleration, velocity, distance travelled, and error messages. To display the various statistics on the screen, we will connect a push button that cycles through the different data. The LCD has a voltage rating of 5V. This display allows for both USB and serial interfaces for communication. We will use the serial interface to avoid the need for a mini-B USB cable. There are three wires needed for the serial connection: black to GND, red to +5V, and white to the 5V TTL serial input. Once connected to the CPU via ribbon cable and pin headers on the PCB, the actual display itself will be mounted on the chassis using bolts and washers.

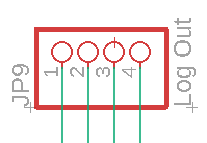
*H. J-Tag Programming Port*

**

The Joint Test Action Group (JTAG) interface is an industry standard that we will use to program the microcontroller. This requires four JTAG pins: TCK, TDI, TDO, and TMS, which corresponds to pins P$2, P$4, P$6, and P$8 respectively (the pins on the left side). To program the CPU, we will be using the Atmel-Ice Basic. On the programmer’s datasheet, the recommended header pinout is a two by five pinout for AVR JTAG. This pinout handles the four necessary pins previously mentioned, as well as the GND (P$1 and P$9 on the right) and VCC (P$3) pins needed to communicate with the CPU. Other pins for the programmer’s pinout are included (NC and TRST), however are not necessary for programming functionality. These pins correspond to pins P$6 and P$8 and are left disconnected on the board.

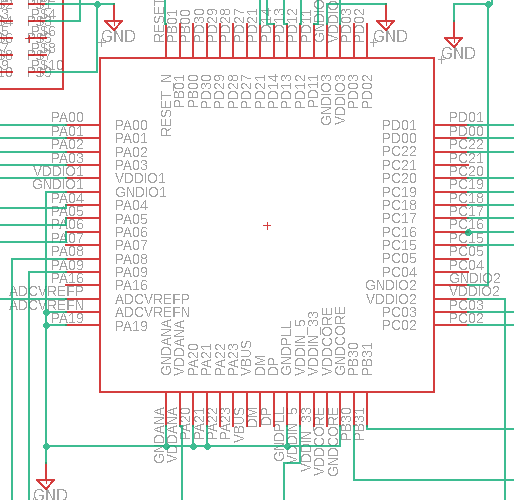
Attempts to program the CPU through the JTAG pin header were met with multiple problems. The programming environment recommended in the processor datasheet returned several errors when trying to upload code onto the processor. For as of yet unknown reasons, the IDE was identifying our CPU as a different device than its part number was. Searches for other working systems using the same CPU as this project revealed that this processor had more niche application, and thus less support, than our initial reading suggested. On top of this mismatched device ID issue, attempts to program the processor with code from the IDE failed with either the IDE being unable to send commands properly or being unable to unset the security bit.

*I. Log Exporting Port*

**

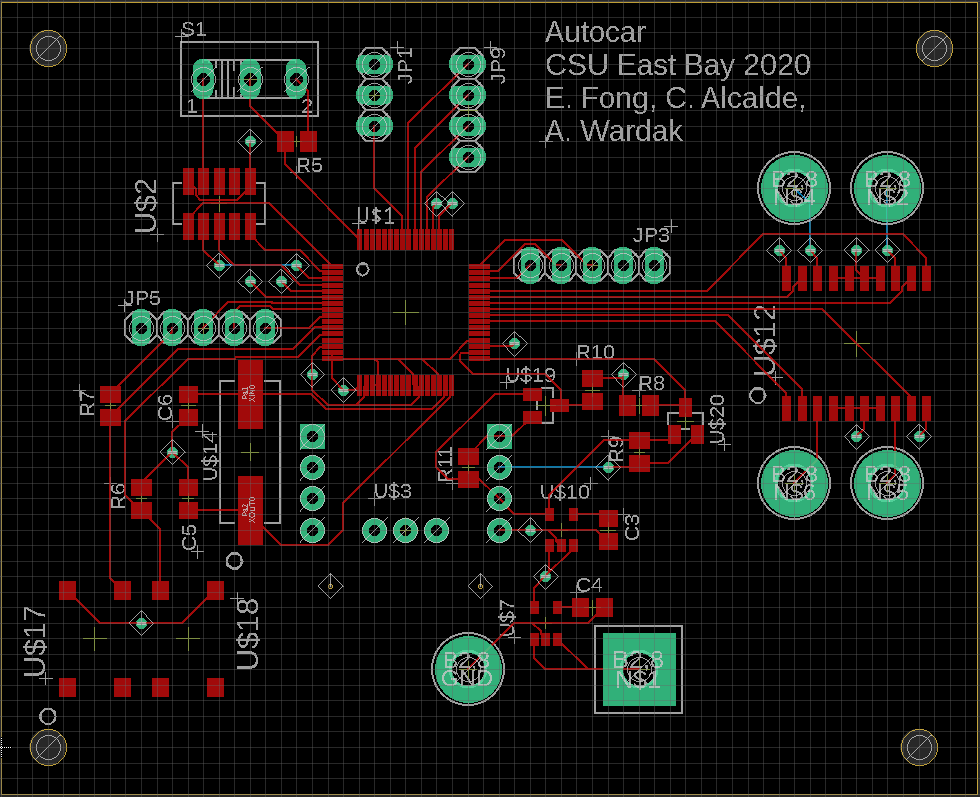
A 4-pin male pin header was implemented on our board for the purpose of our log exporting reach goal. With this pin header, we plan to solder an RS232 shifter. This module, which is essentially a PRT-00449, allows the microcontroller to communicate with a computer via 9-pin UART serial cable. Pins 1-3 are connected to VCC, TX-O, and RX-I respectively, while pin 4 is tied to GND.

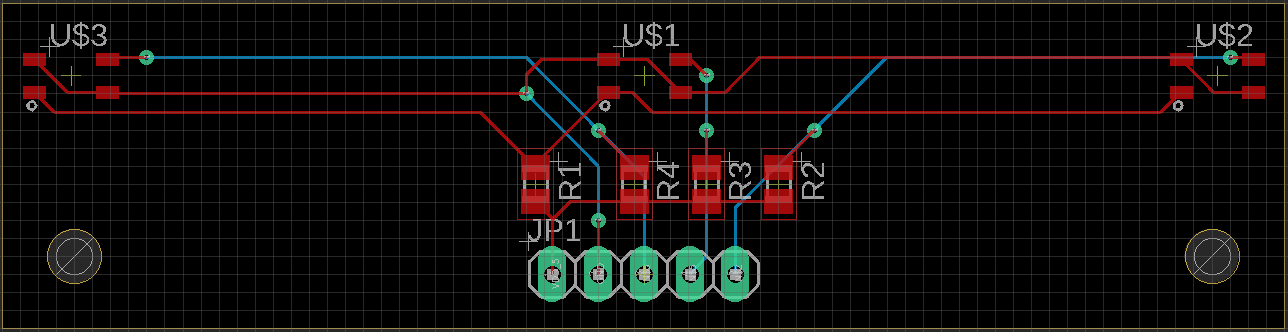
*J. CPU*



The 64-pin AT32UC3C2512C-A2UT-ND is the heart of the entire PCB. Every component is connected or controlled by it in some fashion. Attached on pins PB30 and PB31 is the 32.768 kHz crystal oscillator. Two 10 uF capacitors are also connected to these same pins. The capacitors are also connected to GND. Power is supplied from the 4-AA battery pack to the 5V power regulator to provide +5V. The power regulator is then connected to the various VDD pins of the CPU. This includes pins ADCVREFP, VDDIO1, VDDIO2, VDDIO3, VDDIN\_5, and VDDANA. All ground pins are tied to GND. These pins include GNDANA, GNDIO1, GNDIO2, GNDIO3, GNDCORE, GNDPLL. Other pins, such as PA19 and ADCVREFN are pulled to ground as well. Because there are so many pins on this particular CPU, several go unused. This includes pins PA22, PA23, PB00, PB01, PC04, PC05, PC21, PD27..30, VDDCORE, VDDIN\_33, and VBUS. Though we researched and worked on our board to choose the correct CPU, the programmer encountered some errors causing our group to consider cutting the CPU from the PCB and connect it to a different microprocessor’s (BeagleBone Black/Arduino Uno) CPU. There was one idea about the switch which makes our CPU go to reset mode. Desoldering the CPU was not required but after Professor Tandon’s suggestion, we decided to desolder the PCB’s CPU for further complications of the board.

IV. LAYOUT





Estimated cost of all the board components (main PCB and IR bumper) are estimated to cost around $50 USD. Additional offboard modules such as the Atmel ICE programmer and ultrasonic sensors add more to the total, adding an additional $180 USD. In order to manufacture the PCBs, it would cost $5 USD per IR bumper and $74 USD per main PCB. The total unit cost for both PCB boards and necessary components adds up to a total $306.70 USD (as of May 14, 2020). The total price will increase even more if you decide to have the boards professionally assembled like we did.

Both PCB boards are compact in size, making it easy to situate on a car chassis along with the other off board components.. The main PCB board has measurements of 79.68mm x 64.76mm. The IR bumper board has measurements of 73.64mm x 33.96mm.

Initial Part placement was based on keeping the IMU at the center of the main PCB so as to avoid taking erroneous cardinal movement measurements when in reality the car would be rotating in place. It was later learned that this concern could be worked around by calibrating the IMU for any offset from the center of rotation.

One potential issue with the placement of the crystal in relation to the CPU is asymmetrical wire lengths for each lead connecting the two interfering with the precise timekeeping function of the crystal. Consultation has indicated that part placement and trace routing may need to be redone to remedy this issue.

Our initial JTAG trace routing left the TRST pin unconnected. In troubleshooting programming the processor, we soldered a jumper wire between the JTAG TRST pin and the reset pin of the processor, and flipped the reset switch to the both-disconnected position to see if having the CPU reset constantly pulled to high or low voltage was causing us to be unable to program the CPU. Identical errors showed before and after the modification.

Due to unforeseen circumstances related to the COVID-19 pandemic, our group was met with many challenges that hindered the majority of the semester. As a result, access to the university’s computer engineering lab was cut off, meaning no soldering stations, multimeters, oscilloscopes, or physical meetings with our group members and the professor. Group meetings were now limited to weekly half-hour online meetings with the professor and between group members. This also meant that the handing off of parts was limited and to be done with caution to prevent the possible spread of the virus. Much of the work now had to be done individually because of difficulties with sharing work.

V. SOFTWARE

Several pieces of software were used to help put this project together.

* *Eagle:* Software used to design the schematic and PCB layouts for the main board and IR sensor board.
* *Atmel Studio 7:* Used to program the necessary code and algorithms on the 32-bit AVR microcontroller. The Atmel-ICE Basic served as the physical programmer.
* *Path Follower Algorithm:* Implements three IR sensor modules and executes movement along the blue tape path.
* *Obstacle Detection Algorithm:* Implements two ultrasonic sensors to detect obstructions within 100mm of either sensor. If an object is detected past 100mm, it triggers the circumnavigation algorithm.
* *Circumnavigation Algorithm:* Triggered by obstacle detection. Uses the distance sensors to obtain a “hugging” distance of ~215mm next to an object until the IR sensors detect a tape path..
* *Integration Algorithm:* Integration algorithm using Riemann sums on acceleration data to obtain velocity data, and once again on the velocity data to obtain position data.

VI. MILESTONES

As part of the Senior Design II class, a number of unofficial milestones had to be met in order to help keep track of time and progress for development. Unfortunately, progress on our manufactured PCB was delayed indefinitely because of the unexpected arrival of COVID-19.

* *Milestone #1 – Soldering and power up:* Our group budget allowed for the assembly of five main PCB boards. On one of the boards, we solder two wires to the GND and VDD pads of the PCB so that we could connect the board to a power supply as opposed to the battery holder. Using the power supply, we input 6V into the board and got a reading of 0.02A. Our IR boards had to be soldered by hand. These parts included the three IR sensors, resistors, and wires to connect this board with the main PCB.
* *Milestone #2 –Processor, memory, boot up*
* *Milestone #3 –BIOS level monitor or “operating system”*
* *Milestone #4 –Individual subsystem tests*
* *Milestone #5 –Integration of subsystems*
* *Milestone #6 –Full Application*
* *Milestone #7 –Project demonstration, technical report, and oral presentation*

The COVID-19 pandemic proved to be an obstacle towards making progress, causing each member of the group to work on the project individually. Before the pandemic started, our group was experiencing problems related to the processor. While we were able to power up our processor, booting up was a problem. In Atmel Studios 7, the processor was being identified as a different processor model. Although we tried to look past this, we kept getting error messages that prevented us from programming the processor. With the professor’s help, we decided that the problem might have been related to the JTAG reset pin not being connected to the processor’s reset pin. Despite soldering a jumper wire between the two connections, the same error kept on showing up.

This problem ran several weeks into the semester with no solution in sight. Because of COVID-19, it was hard to find solutions to the problem because although we each had a PCB, there was only one programmer available. Essential tools to help us debug the board (multimeter, oscilloscope, soldering station) were not available to use unless we paid out of pocket. Had there been no pandemic, our group would have opted to respin the board with a different processor. With approval from our instructor, each group member purchased a microcontroller board to provide a proof-of-concept of our project. In theory, by getting each subsystem to work on our microcontrollers, implementing them to our PCB would be easier since we would just need to translate the code to the respective processor.

Despite this, each group member chose to work with either an Arduino Uno or BeagleBone Black. Using the resources that we had, our group was able to implement some of the subsystems in our autonomous car. Using our microcontrollers, we were able to show proof of concept for obstacle detection. By connecting two of the ultrasonic sensors, one of the members was able to record the distances between various objects. Because our original design used two ultrasonic sensors, we had also tested a second sensor to work alongside each other. We were also able to partially implement the LCD character display in the design by getting the display to show the input of both distance sensors simultaneously.

Drivers for the ultrasonic sensors, H-bridge, IR sensor array, and LCD display were successfully written for use with the Arduino Uno and Beaglebone Black microcontrollers. While our implementation of the level shifter between 5.0 V and 3.3V was functional, development of drivers for the accelerometer was not achieved.

VII. REACH GOALS

While coming up with the initial design of the autonomous car, we wanted the car to be able to follow the blue-taped line as well as avoid obstacles that may be in the way of the vehicle. Our reach goals included implementing a more advanced obstacle reaction algorithm. Some of these features include accurate obstacle readings while the car is in motion, detecting if an obstacle is in motion, a “back up” from obstacle function, and an option to wait for traffic to pass. We have also included an extra serial pin header on our main PCB to implement our reach goal of a travel log export via serial UART cable. If given more than a semester to complete the project, the Engineers believe that these stretch goals could be implemented along with the car’s other basic functions.

VIII. CONCLUSIONS

In any engineering field or project, the goal is not to fail but to succeed. Due to a lack of experience and documentation, our group encountered some difficulties. The most important thing that we learned from our project was that choosing a non-popular part is not a good idea. The processor that we used, the AT32UC3C2512C-A2UT-ND, had a problem when we started programming it. Due to the limited amount of time we have for the project, using a different microcontroller’s CPU for our project is very essential for now. With the unfortunate timing of the COVID-19 pandemic, our work towards this project was interrupted. At the very least, our group was able to power up the main board. Had the option been available, our group would have chosen to respin the board with a new processor that has more documentation and industry usage. Doing this would allow us to move forward instead of circles.

Our group made some discoveries while using our subsystems that would allow us to design improved subsystems in any future iterations of this project.

We found that the ultrasonic sensors had reduced reliability in determining the distance to an object when said object had a surface that dampened sound waves, like cloth or foam. This would lead us to consider alternative proximity sensing technologies such as capacitive, optical, or other types of proximity sensor, and possibly using multiple types of proximity sensor for decreased environmental interference and increased resolution and range.

During testing of our motor control operations using the H-bridge, we observed that said operations introduced a significant amount of noise into our power subsystem. This issue certainly would have resulted in many more erroneous measurements from the accelerometer and timekeeping crystal. To rectify this, we would separate the power subsystems for the motors and our more noise-sensitive devices.

Initial layout of our PCB was mistakenly focused on the idea that the IMU should be placed as close to the center of the board to avoid taking erroneous measurements such as detecting a forward or sideways linear acceleration from a rotational movement. We later learned that we could calibrate our IMU to nullify this concern, however this misconception resulted in the nonoptimal placement of some of our other circuit components. Most notably, the crystal had asymmetrically sized and shaped traces to the corresponding pins on the CPU. This would have resulted in less reliable timekeeping for our CPU clocking and IMU measurements and calculations.

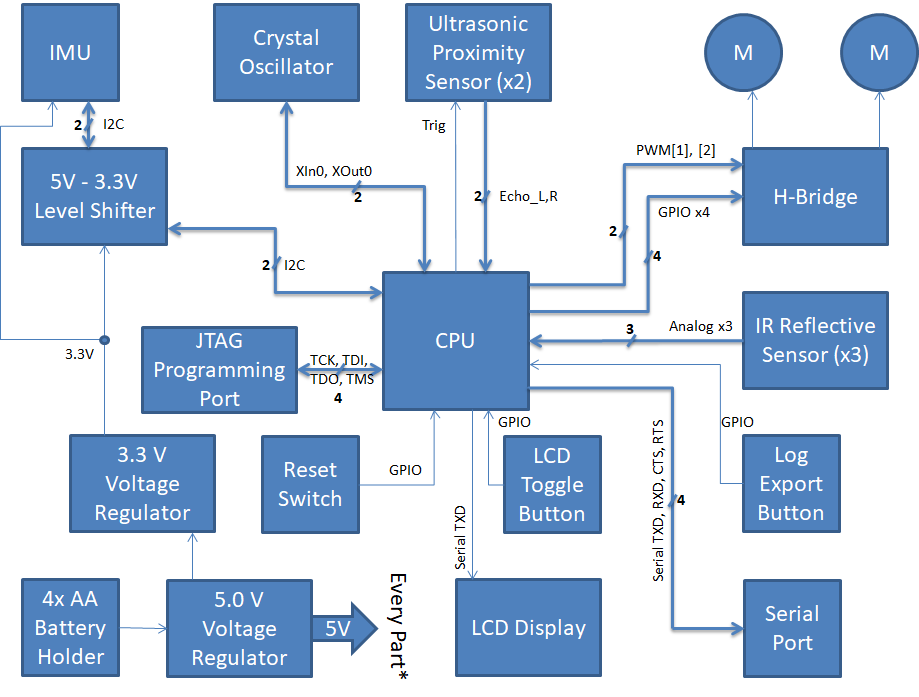
In the end, our group was able to produce a proof-of-concept prototype using Arduino Uno and BeagleBone Black microcontrollers. If this project were to be revisited, we hope to turn the proof of concept into a more fully-fledged prototype.

IX. BIBLIOGRAPHY

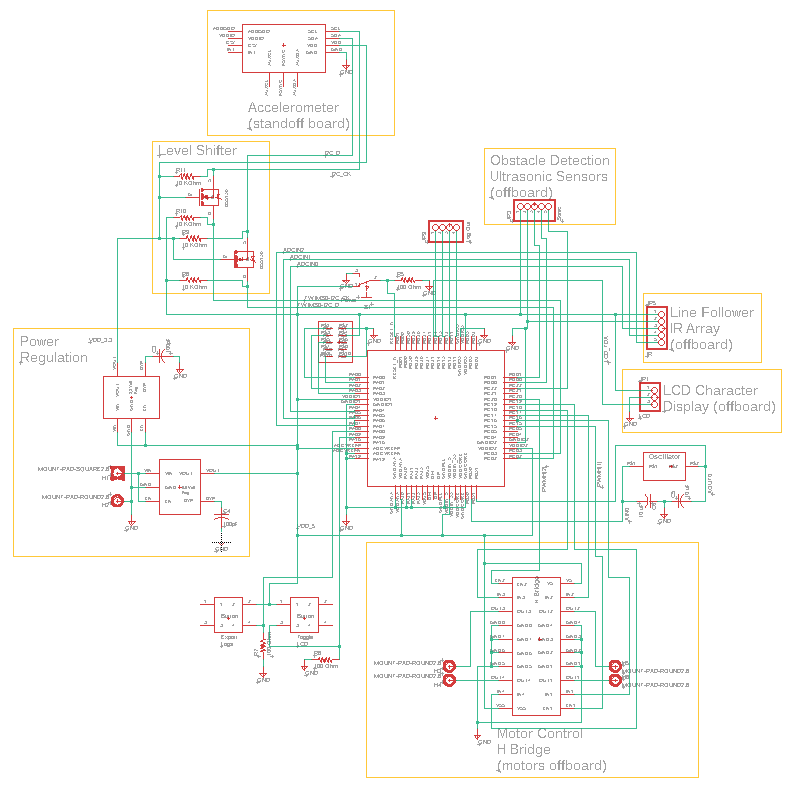
* *32-bit AVR® Microcontroller*. (2012). Microchip Technology. Retrieved April 10, 2020, from https://ww1.microchip.com/downloads/en/DeviceDoc/doc32117.pdf
* *AP2210 Series*. (2016, June). Diodes Incorporated. Retrieved April 10, 2020, from https://www.diodes.com/assets/Datasheets/AP2210.pdf
* *PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES*. (2003). STMicroelectronics . Retrieved April 10, 2020, from https://www.st.com/content/ccc/resource/technical/document/datasheet/04/ac/22/f9/20/5d/43/a1/CD00000059.pdf/files/CD00000059.pdf/jcr:content/translations/en.CD00000059.pdf
* *LFXTAL003210Bulk Datasheet*. (n.d.). IQD Frequency Products. Retrieved April 10, 2020, from https://www.iqdfrequencyproducts.com/products/pn/LFXTAL003210Bulk.pdf
* *BSS138 Datasheet*. (n.d.). ON Semiconductor. Retrieved April 10, 2020, from https://www.onsemi.com/pub/Collateral/BSS138-D.PDF
* *MPU-9250 Product Specification Revision 1.0*. (2014, January 17). InvenSense Inc. Retrieved April 10, 2020, from https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-9250-Datasheet.pdf?ref\_disty=digikey
* *Miniature Reflective Object Sensor*. (n.d.). ON Semiconductor. Retrieved April 10, 2020, from https://www.onsemi.cn/PowerSolutions/document/QRE1113-D.PDF
* *Ultrasonic Ranging Module HC - SR04*. (n.d.). Elecfreaks. Retrieved April 10, 2020, from https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

X. APPENDIX

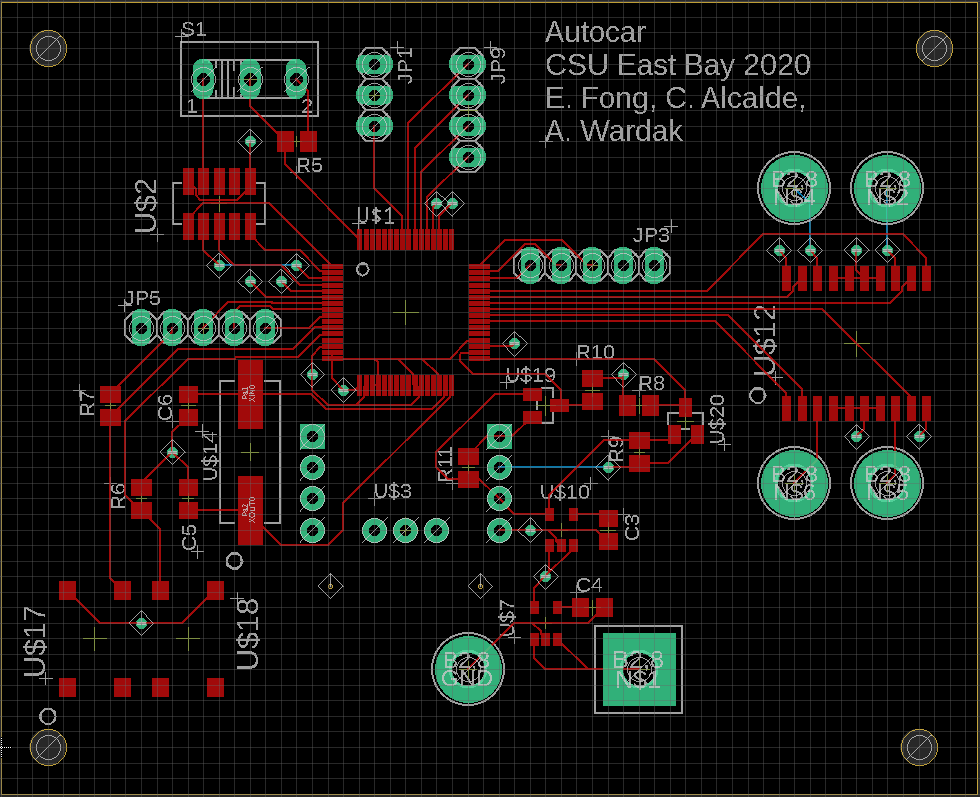
1. *Top Level Block Diagram*



1. *Main Board Schematic*



1. *Main Board PCB*



1. *IR Sensor Bumper PCB*

