CECS 347 - Final Project

Obstacle avoiding vehicle with sharp IR distance sensors and LCD

By

Zachery Takkesh & Justin Maeder

May 9th, 2019

Description: Create an autonomous vehicle

that can navigate an obstacle course

using two sharp IR distance sensors and displaying with the Nokia 5110 LCD.

Introduction:

For the final project in our Microprocessors and Controllers II class, we are to utilize our knowledge of C language for software and hardware to modify and improve our previous vehicle from project 1. In our first project, we used the two DC motors and the TM4C123 onboard LEDs to show the speed and direction of the robot vehicle. The hardware we added for this project included two sharp IR distance sensors and an LCD. The sharp IR sensors are mounted at different on the front of the vehicle and serve a purpose to send information when an obstacle comes in range, and the LCD will display the obstacle distance information in centimeters combined with the Pulse Width Modulation cycle or PWM. The last piece of hardware in our design is the 10k ohm potentiometer which allows us to control the PWM cycle and controls the speed of the vehicle. Overall, we are modifying the robotic vehicle in this project to be able to navigate around objects autonomously.

Operations:

The most significant learning curve of this project was the ADC or analog to digital conversion. Two other operations used were the Pulse Width Modulation cycle and the synchronous serial interface on the Blue Nokia LCD. Other hardware components include robot chassis, two DC motors, 12V battery pack, H-bridge driver, two sharp IR sensors, and a potentiometer. The 10k ohm potentiometer resistance value initializes the speed of the two DC motors. This analog potentiometer value is used by software to convert the ADC value to the corresponding PWM cycle value. When the ADC value is 0, the DC motors will be stopped. When the potentiometer ADC increases, the PWM increases, and the DC motors start to spin when the PWM reaches 15%. If the ADC is increased further, the PWM can reach up to 98%

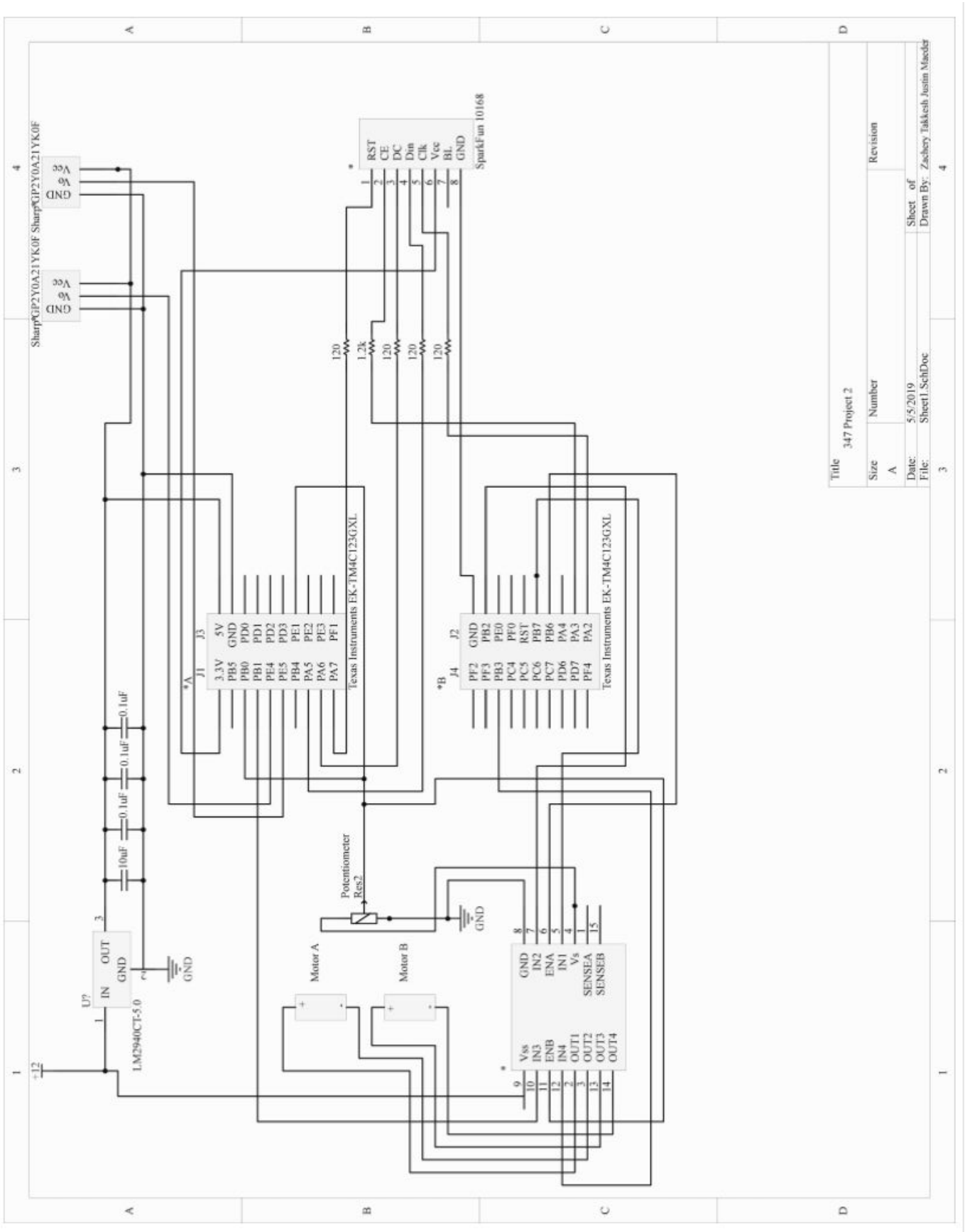
PWM with a 3.3V output. The two IR sensors that are attached in front of the chassis output an ADC value that is input to the software. The sensor ADC value is converted to distance in cm to accurately depict how far away an obstacle is. This is updated every 1 microsecond so the robot vehicle can avoid any objects in its path by lowering the PWM of the opposite wheel to turn out of the way before a collision occurs. Lastly, the LCD is used to display the distance of each of the two sharp IR sensors (left and right) in centimeters and the potentiometer PWM duty cycle. The software used to construct and control the autonomous robot vehicle is in the language C using the Keil uVision4 compiler and interfaced with the EK-TM4C123GXL Arm Cortex Microcontroller.

Theory:

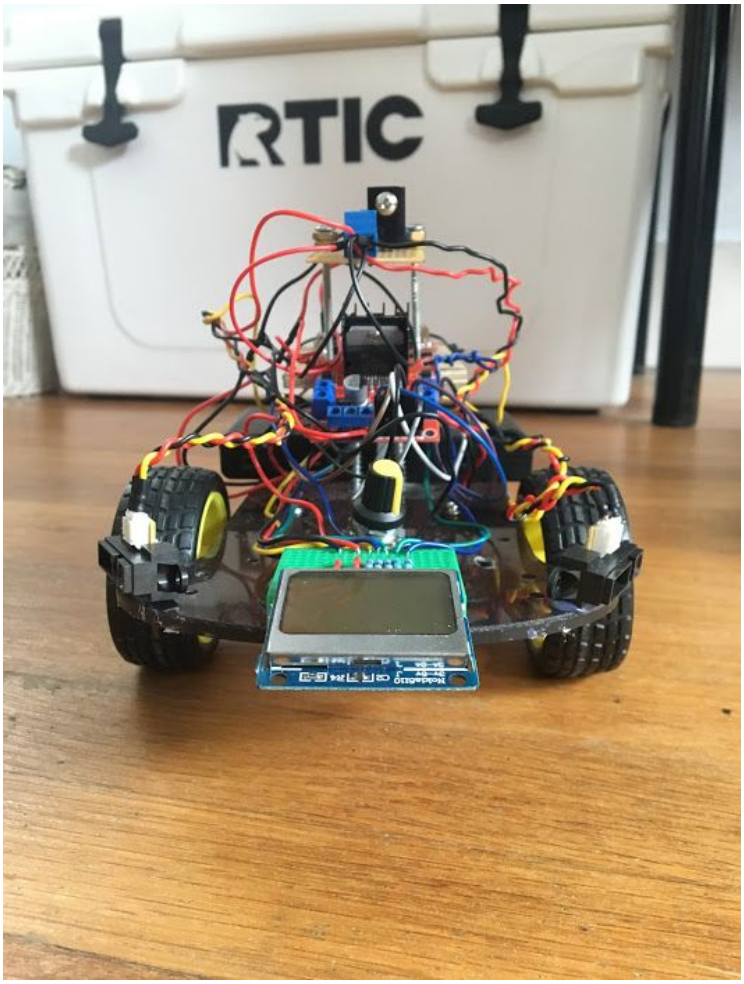
This project assists with demonstrate the understanding of interfacing hardware using analog components with a Tiva-C TM4C123GXL microcontroller and accurately implements the concept of pulse width modulation(PWM), synchronous serial interface(SSI) with the Nokia 5110 LCD, analog to digital conversion(ADC), and wiring other hardware components together like the H-bridge, potentiometer, and designed power supply. The microcontroller is interfaced with two Sharp IR sensors, and the LCD is used to show the pulse width modulation duty cycles obtained from the Sharp IR sensors. The PWM generates an output wave of a fixed frequency which is the duty cycle. The potentiometer is used to modify the duty cycle from 0 to 100% for the power driven to the DC motors. The output voltage from the potentiometer ranges from 0 to 3V. When the potentiometer is at 0V, the DC motors are not driven and thus stationary, while the voltage increases, the DC motors speed will increase as more voltage is driven to each motor. For this project, the max speed of the DC motors is set to a 98% duty cycle to ensure the DC

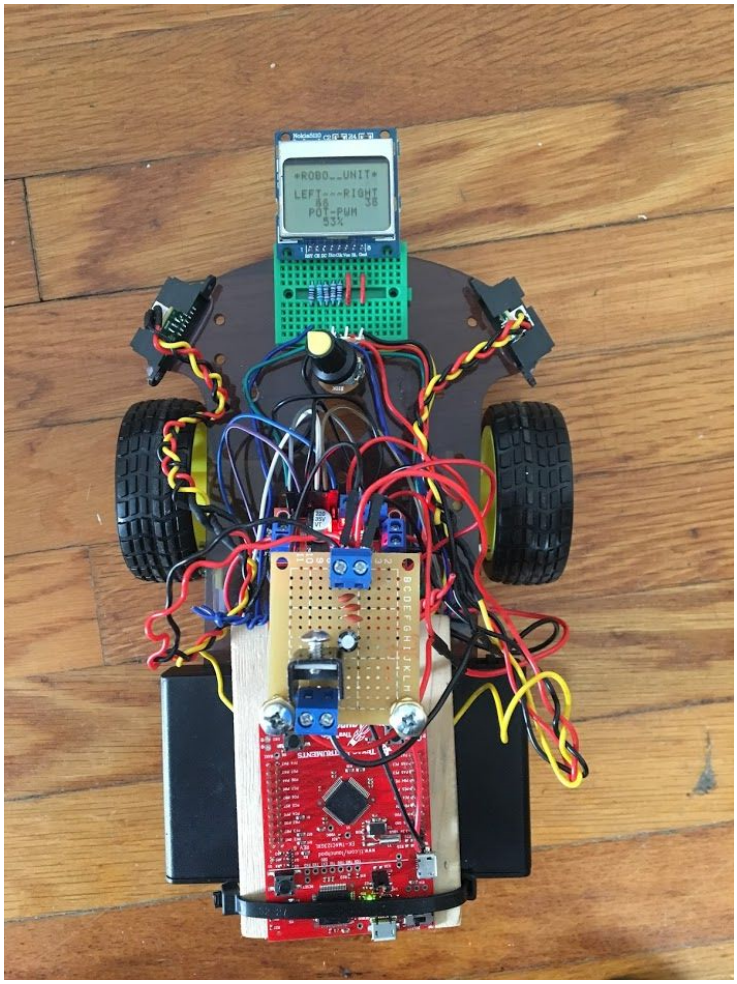
motors are not damaged from the excess current as well as having a good speed to run the car. The TM4C123G's ADC collects and samples data using a sample sequencer. The sample sequencers have the same functionalities except for the number of samples they can collect. Our car robot uses sequencer 1 with the four samples and the maximum sampling rate at 125k samples per second. Port E is initialized to receive ADC values coming from both Sharp IR sensors to calculate the distance based on the data. To precisely convert the ADC values to the distance we collected data from physical testing of the sensors and computed an equation for the ADC to distance formula. The ADC, as well as the voltage values from a DMM, were recorded starting from the distance of 5 centimeters to 70 centimeters away from the sensor with 5-centimeter increments. The recorded values from the experiment are saved in a list contained within the code. The synchronous serial interface is a synchronous serial communication channel for digital data transmission from components to the microcontroller. One of the SSI channels is used to control the Nokia 5110 while port A is used to drive the output values for the LCD. The Port A pins 2, 3 and 5 are used for the synchronous serial communication to the LCD. The LCD will display the duty cycle of power being driven to each motor based on the readings of the Sharp IR sensor as well as the potentiometer reading.

Hardware Schematic:



Hardware:





Software:

The potentiometer and two sharp IR sensors (left and right)are the three analog-to-digital-converted inputs that are implemented to control the robot vehicle with port pins PE1,5, and 4 respectively. For our case, we are using ADC0 sequencer 2 to complete the task which is initialized with a 125k samples/second rate. The systick handler initiates SS2 then reads the potentiometer, right sensor, left sensor, then proceeds to clear the flag. The two ADC sharp IR sensor signals are used in the equation (-15.43 * log(adc_value) + 124.13) to calculate the distance of the object as the robot vehicle approaches it. For this equation, y is the distance,

and x is the ADC value per sensor. The distance from each sharp IR sensor is then updated and displayed on the Blue Nokia5110 LCD. Below the two sensor values is the PWM duty cycle percentage that is determined by the potentiometer reading which ranges from 0-98%. This percentage is calculated by taking the sharp IR sensors ADC value and multiplying it by 100 then dividing by the max ADC value (4095). This calculated percentage is used to determine the PWM duty cycle of the two DC motors through the port pins PB6 and PB7. To control the car's movement, we created a function with five separate conditions. First, we initialized both wheels to turn in a forward direction. The first condition was for the worst case scenario if both left and right sensors are less than 10 cm, then both wheels will go backward until out of range to re-navigate. Second, if the left distance is less than 22 cm, then the left wheel will turn forward, and the right wheel will turn backward to navigate towards the right. Third, if the right distance is less than 22 cm, then the left wheel will turn backward, and the right wheel will turn forward to navigate left. Fourth, if both the left and right sensors are out of range (33cm or greater), that means the robot vehicle has reached the end of the track and will stop. Lastly, if the first four conditions aren't met, both wheels will turn in the forward direction at the given PWM cycle.

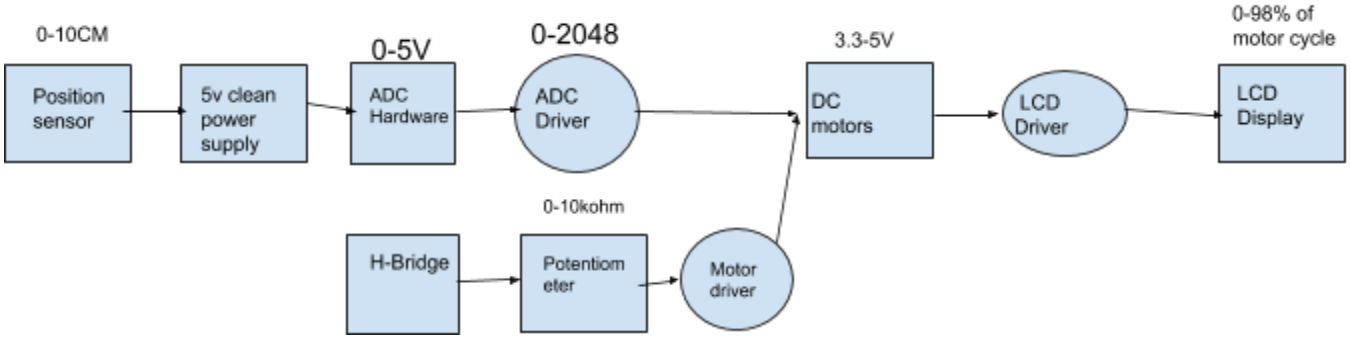
Conclusion:

When designing and programming all of the additional features of the robot, there were many software and hardware issues that we dealt with. One of the big problems we encountered was involving the Sharp IR sensors. Both of the sensors required a voltage input, ground, and output ADC values into the board. Pins PE 4 and 5 are used to received the outputted ADC values while computing the distances. The problem was that the ADC values were not stable enough which then caused the calculated distances to be inaccurate and the calculated displayed

distance to change every retrial of the experiment. To combat this issue, we ensured the voltage going into the Sharp IR sensor was stable and consistent using a more exact formula with the code. To stabilize the voltage we used multiple capacitors and an NTE 1951 voltage regulator which converted a 12v input to a 5v output. This designed power circuit regulator produces clean and stable power for the IR sensors as well as powering board. A software issue was found when reading the ADC value, which resulted in the distance measured by the sensor being off by at least ten centimeters. To fix this problem we tried various formulas that would give users the most consistent and accurate reading. We implemented a power formula to find the best-fit equation based on the values we recorded for the ADC test. Another issue we encountered was the battery power supply which powers the whole car robot, and its components would slowly drop in outputted voltage which in return would cause the components to regress in performance and accuracy. To resolve this issue, we merely replaced the lower voltage batteries with fresh 1.5v grade batteries. Another issue we came across was the correctness of the sensor reading when it came to demonstrating the car on the track. The IR sensors would pick up the wall when the car was not close to the track wall, thus making the car turn when it was not supposed to. To combat this issue, we had to resample and adjust the ADC reading values to ensure the sensors work when supposed to. Another issue we had was the grip on our wheels were too sticky and caused turning issues when detecting a wall. To fix this issue we implemented a feature to our car that made the car reverse a few centimeters and turn one wheel with more power and the other wheel with less to implement to the motion of a car turning.

Demo Link: https://youtu.be/eNqPrMl8qbo

Data Flow Graph:



Call Graph: