CECS 347 SPRING 2018 Project 3

WALL FOLLOWING ROBOT

By

Michael Diep & Steven Sallack

05/09/2018

This project builds a robot car that uses three IR sensors to detect obstacles and navigate itself through an obstacle course.

Introduction

In this project we are designing a robot car with motor control and analog sensors. To power the DC motors we will be using an H-Bridge. The H-Bridge provides direction and speed/power control. The car can move forwards and turn left and right at varying speeds depending on obstacles detected. The LCD display connected to our TM4C board will display the 3 sensor distance (cm) outputs as well as the PWM Duty Cycle percentage.

Operation

The robot car's wheel speeds are adjusted by detecting how close it is to a wall on either side. If it is close to a wall on the left side, the left motor will speed up and steer the car back towards the center of the course. The same logic applies for the right side. The sensors will output "out of range" to the LCD screen if no object is within detection distance, otherwise they will display their distance from the object in centimeters. The robot car will be able to determine which distance is greatest and which distance is the shortest. There is also a front sensor to detect when we are in danger of running into an obstacle head-on. It is used to execute faster turns than it would otherwise perform.

Theory

PWM – for this implementation, we used hardware PWM to control the speed of our motors. We referenced the PWM starter project which required little to no changes as we were able to use the default pins (PB7, PB6) in our design. It is important to note the details of this PWM implementation such as which module and generator were used, the duty cycles available, and how the duty cycles can be changed:

-The duty cycles available range anywhere from 0-100%, although some restrictions are encountered due to motor functionality. The more realistic range is from 25%-99%.

-There are 2 PWM modules with 4 PWM generators each. Each generator can generate 2 output PWM signals. These 2 signals will have the same duty cycle and frequency. PB6 and PB7 PWM signals used in this project are from Module 0, Generator 0. We use the same generator for both signals because we want the signals to be the same, otherwise the motors would not turn at the same speed and not go straight.

- We can change the duty cycle by altering the COUNT value in the PWM module.

H-Bridge – We used a different version of an H-Bridge in our implementation that allowed us to source more current to our motors. It has a maximum current potential of 2A with 3A peaks. The motors in our design only pulled 100mA so we were well under the limit. The H-bridge has a power supply input and a ground. The power supply input recommends a 7-25V supply. It has 4 output pins for the 2 DC motors. It has 6 GPIO input pins which control the direction and motor enable. Our PWM signal goes to the motor enables in order to control the speed of the motors. The other 4 pins used to control direction are in 2 pairs for each motor. The 2 pins in each pair need to be inverted, either 10 or 01. The inversion of these bits will change the polarity of the output to the motors.

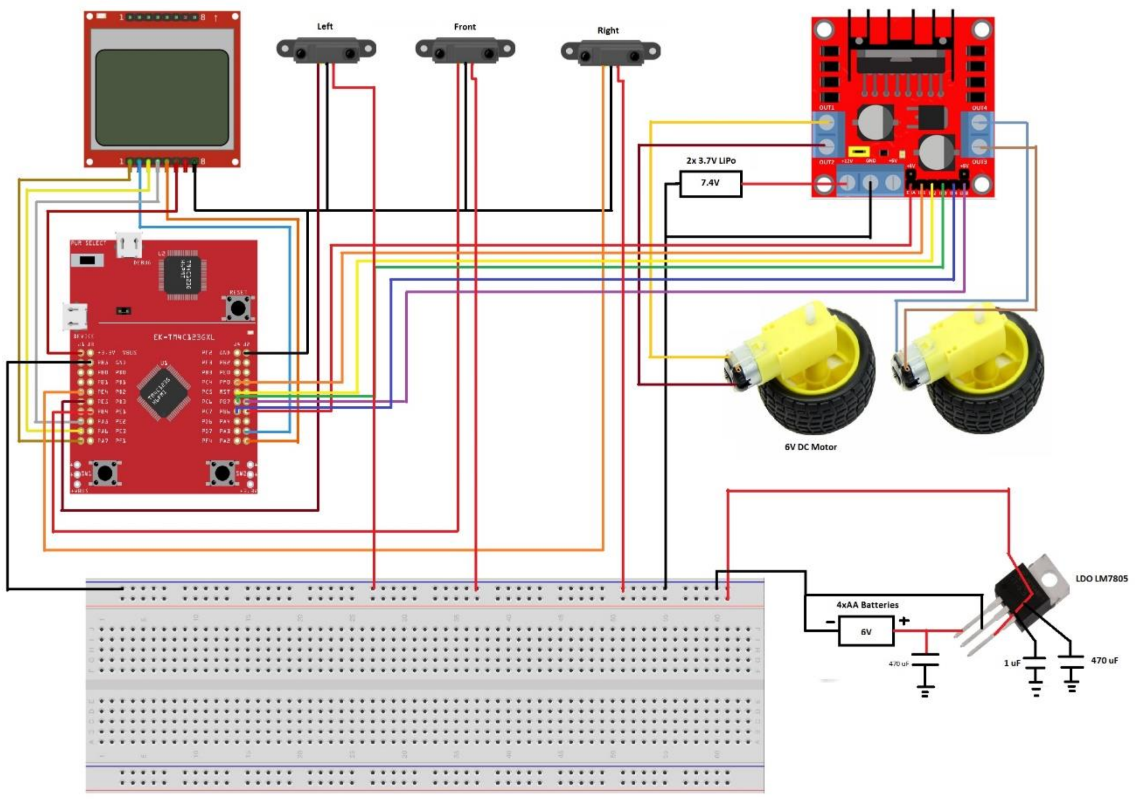
Voltage Regulation Circuit – In this project we had to have power supplies on the robot car. In order to power the TM4C Microcontroller we had to use a voltage regulator circuit. This circuit consisted of an LDO Voltage Regulator and 3 capacitors to smooth the input and output voltage (1x470uF on the input, 1x470uF and 1x1uF on the output). The 470 uF cap on the output smooths out large peak/low frequency changes in voltage and the 1 uF cap smooths out the high frequency, low peak change in voltage. This is necessary to make sure we do not burn out our microcontroller. We use a 12v pack of AA batteries to power the circuit.

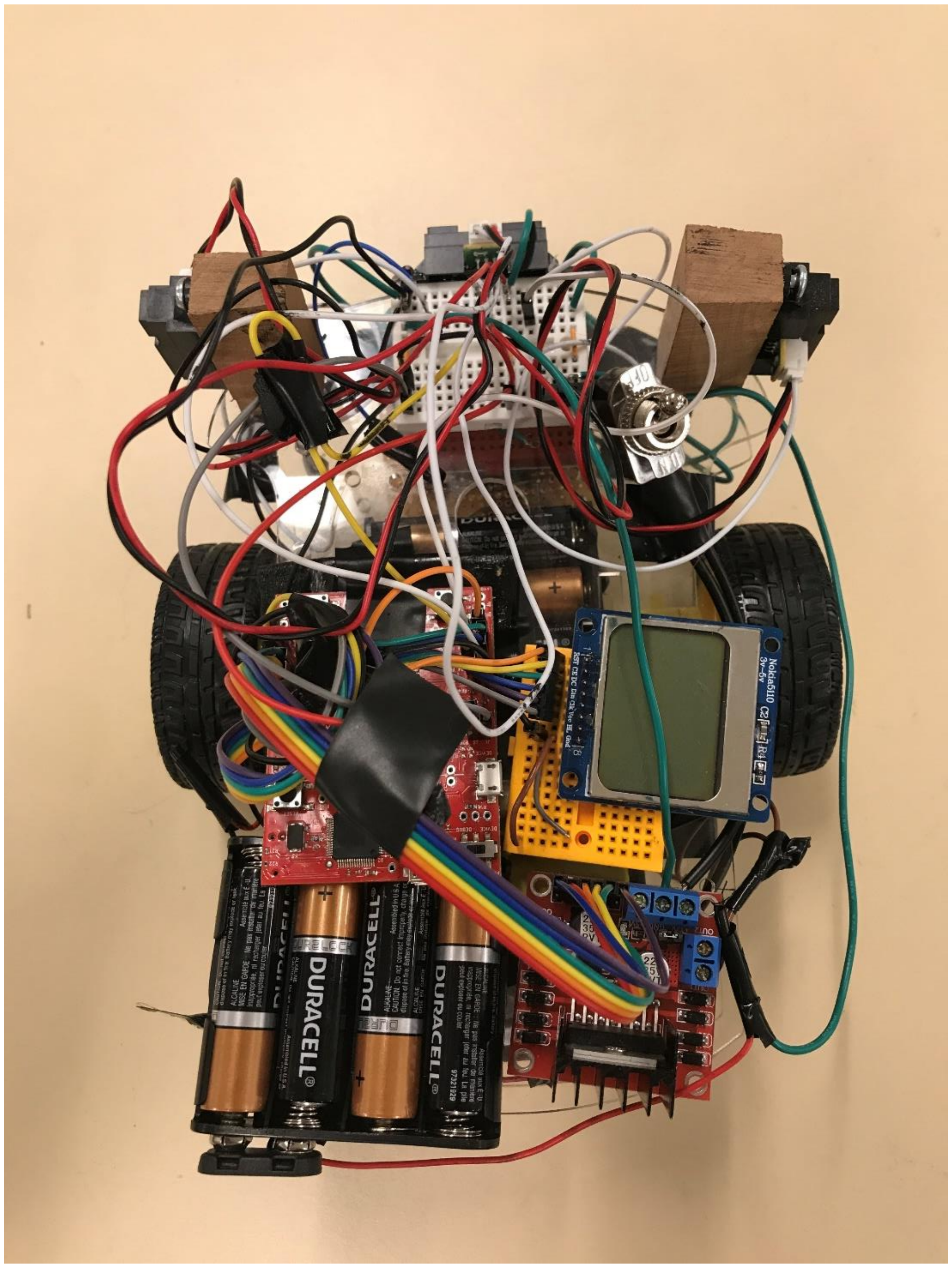
ADC (Analog to Digital Converter) – In this project we are implementing the use of the ADC. It is used to convert an analog input into a digital signal that can be processed and used in the implementation of our overall design. The TM4C has two 12-bit ADC's that are identical. There are 12 pins on the board that can handle ADC analog signal inputs. The two ADC blocks have 4 sequencers, each varying by the depth of their FIFO and the number of samples they can convert. We will be using sequencer 2 for this project, which can handle up to 4 samples. We will be using it to sample the 3 analog distance sensors. SysTick timer will be used to cause the ADC to convert its samples at a frequency of 40Hz. 40 Times per second, the LCD display will be updated with the new value of the ADC outputs and the PWMs will be updated with their new duty cycle values, if the sensor logic determines there is a need for them to be altered.

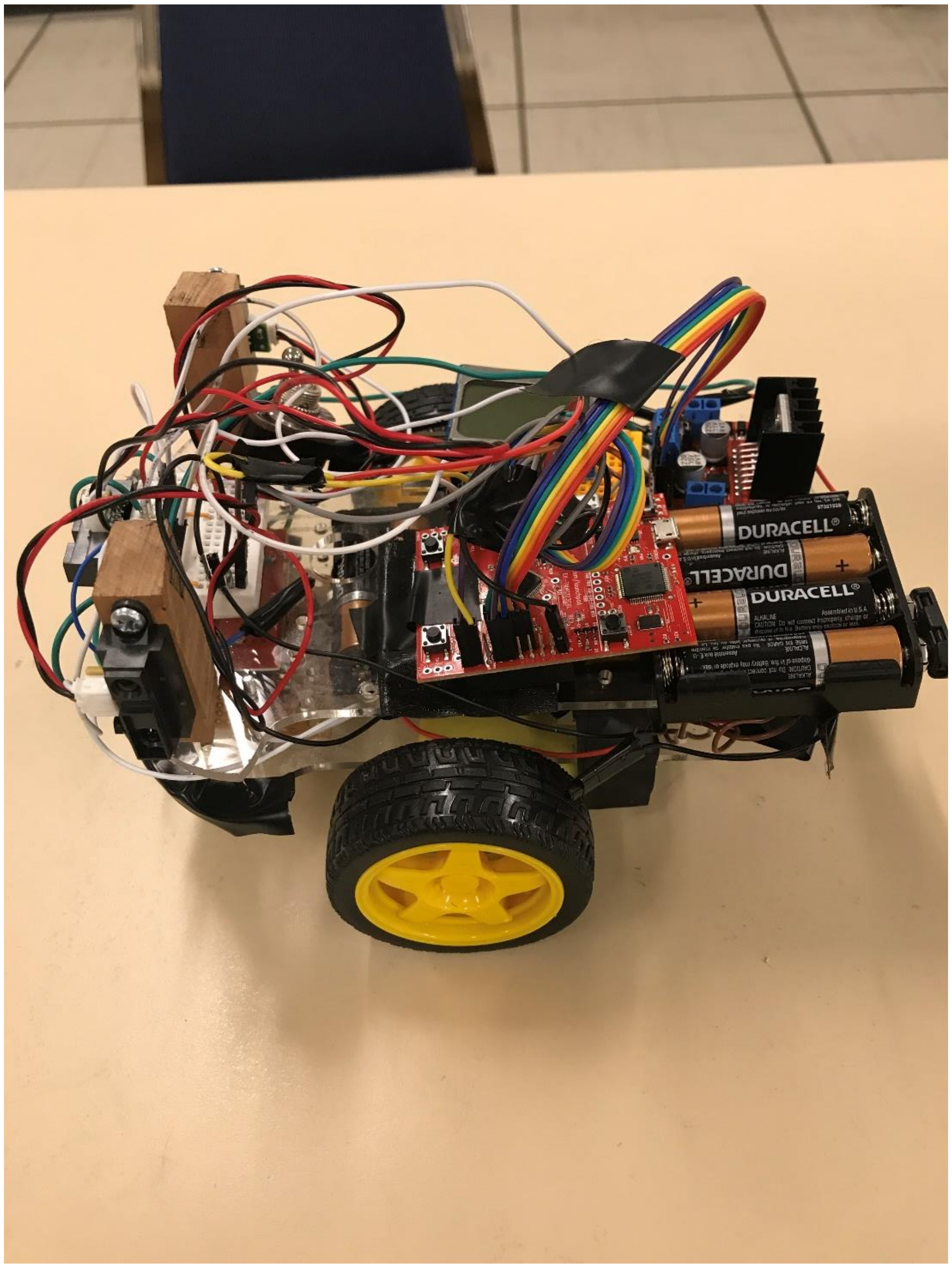
Sharp IR sensor (analog distance sensor) – This sensor is used to detect the distance that our robot car is from any obstacles in their path. It is an analog sensor, operating voltage is 4.5v~5.5v, output voltage is 0~3v, resulting in a 12-bit ADC conversion able to represent 4096 values at a resolution of ~733uV. We will be using a formula, manually generated, to predict the distance of the object from the sensor based on the voltage output of the sensor. There are some potential complications with the sensors, such as direct sunlight, tungsten light sources, noise, etc. Software filters will be used to reduce noise complications, and care will be taken to avoid light-pollution.

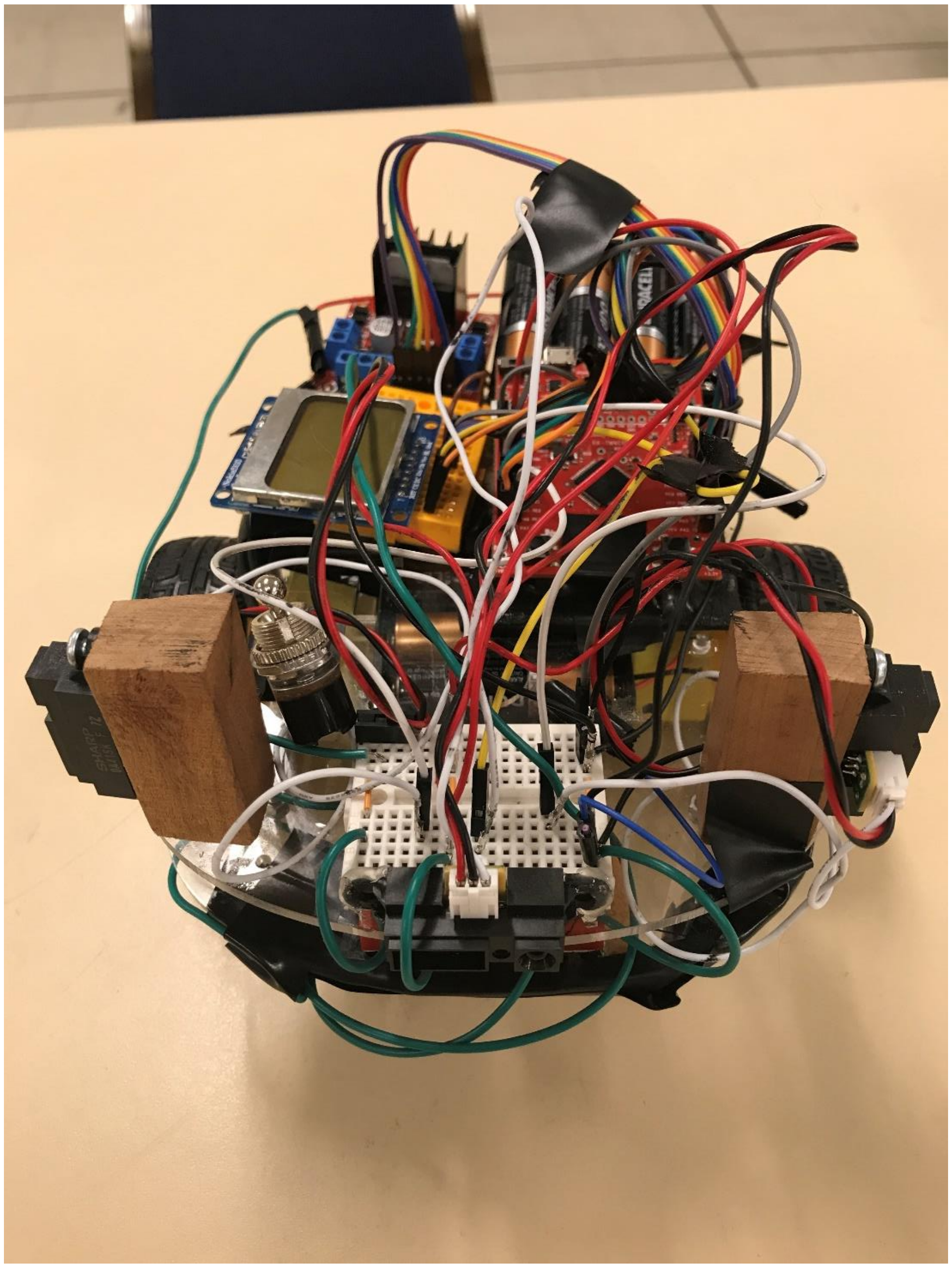
SSI (w/ Nokia 5110 LCD Display) – SSI is a serial communication interface we are using to communicate with the LCD display. It is operated using a common clock between the two sources. It can both transmit and receive, but we will only be using the transmitter because we do not need data coming from the LCD display. The SSI uses a master-slave methodology where the master commences any data transmissions and the slave responds to them. The SSI can send either DATA or COMMANDS. The commands are used to program the SSI and, in our case, the backlight brightness of the LCD display. The SSI is used by writing data to SSIx_DR_R register. The internal hardware components will take care of sending the data out, but care must be taken to check the BSY and TNF flags when sending data through.

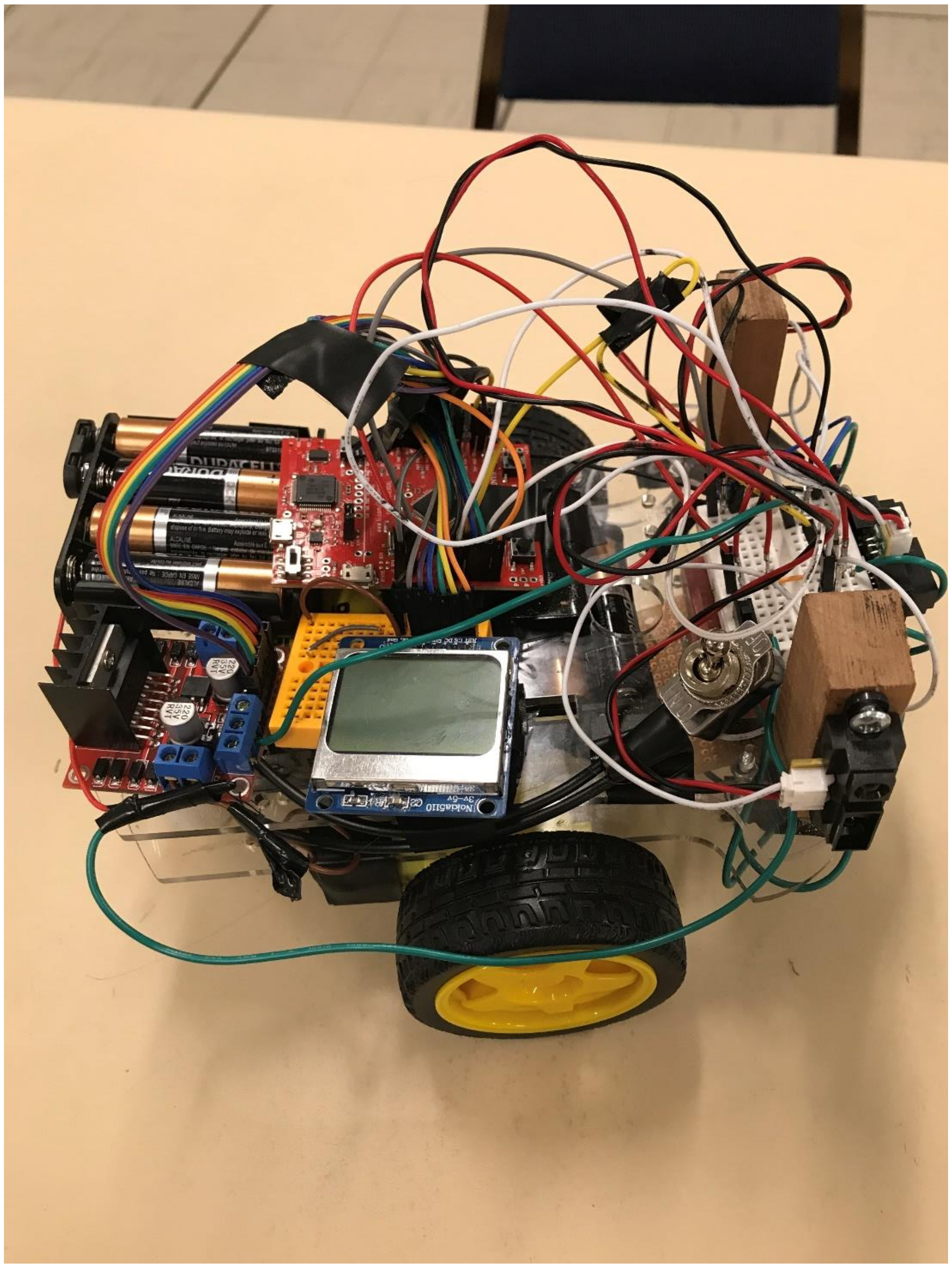
Hardware Design





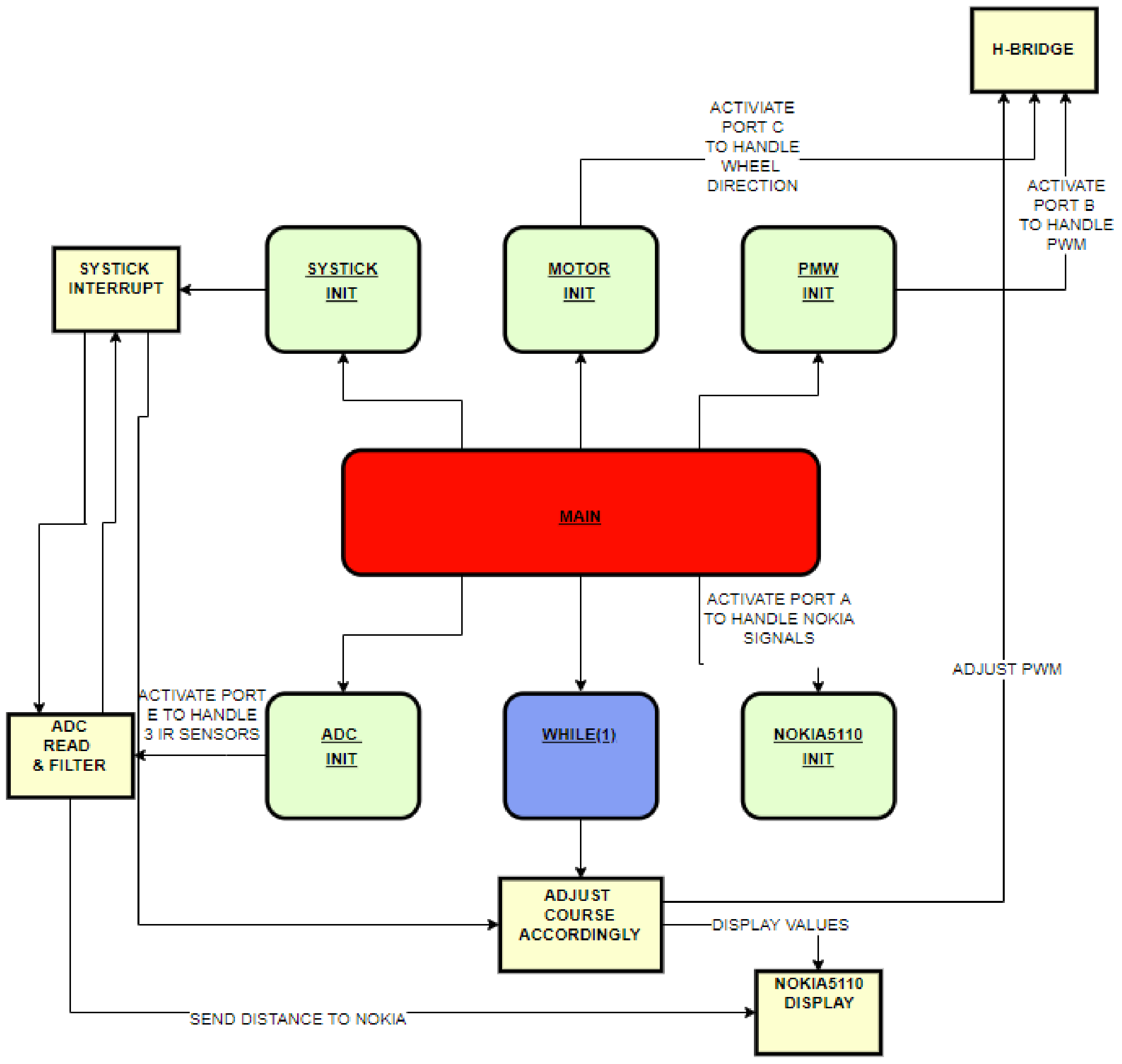

DURACELL
DURACELL
DURACELL





Software Design

To start this project we were given a working sample code that controls hardware PWM which used 2 signals from Port B. For the input to our H-Bridge we used 4 bits from Port C, 2 for each motor to control the direction. To handle the constant change in the IR sensors, we used a SysTick interrupt with a frequency of 80Hz. To input the values of the IR sensors, we used a given ADC algorithm to filter and median the data coming in. We also used 8 pins from Port A to intialize the Nokia5110.


H-BRIDGE
ACTIVIATE PORT C TO HANDLE WHEEL DIRECTION
ACTIVATE PORT B TO HANDLE PWM
SYSTICK INTERRUPT
SYSTICK INIT
MOTOR INIT
PMW INIT
MAIN
ACTIVATE PORT A TO HANDLE NOKIA SIGNALS
ADJUST PWM
ACTIVATE PORT E TO HANDLE 3 IR SENSORS
ADC READ & FILTER
ADC INIT
WHILE(1)
NOKIA5110 INIT
ADJUST COURSE ACCORDINGLY
DISPLAY VALUES
NOKIA5110 DISPLAY
SEND DISTANCE TO NOKIA

For the flow of this code. We start by initializing of our pins used (green boxes). Next, we head into our super loop. Since our SysTick interrupt is occuring at a frequency of 80 Hz, it constantly updates the distance values of the IR sensors. It does this through our ADC algorithm that filters our ADC values. In the SysTick interrupt is when the distance values are calculated using our own calibrated formulas. In our main, everytime the loop reiterates we are comparing the distance values to pre-set values that determine if the robot needs to adjust itself or not. The robot will then adjust accordingly, and display the new IR distance, PWM, and error values on the Nokia 5110 display

Conclusion

The first problem we ran into in our design was that we did not enough power to run the motors through the H-Bridge. We started with 6V, but the H-Bridge requires at least 7.4V to run the two 6V DC motors. In order to fix this ,we sourced the motor power directly from two 3.7V LiPo batteries in series. At full charge, those two batteries left us with about 8.3V and was sufficient to power the motors.

The second problem we ran into was not properly setting up the common ground. When we added the second power supply, we did not connect the grounds together, so the design did not run properly.

An additional problem we found was that the sensors were not getting enough voltage. We started off using a 6v battery pack but we found that the voltage regulation circuit was soaking up 1.5v and we were only left with <4.5v to power the sensors. This created buggy sensor readings. We fixed this problem by adding an additional 6v battery pack in series with the original, creating a 12v battery pack to power the 5v regulation circuit. This solved our problems with powering the IR sensors.

The final hardware problem was an issue of cross-talk. The sensor readings that were analog signals going to the IO pins on the TM4C were picking up interference and leading to skewed distance readings. We eliminated the cross talk by routing our wires around the board and away from any potential noise causing circuitry and connections. Any other problems were related to software and will be detailed in the software description in this report.