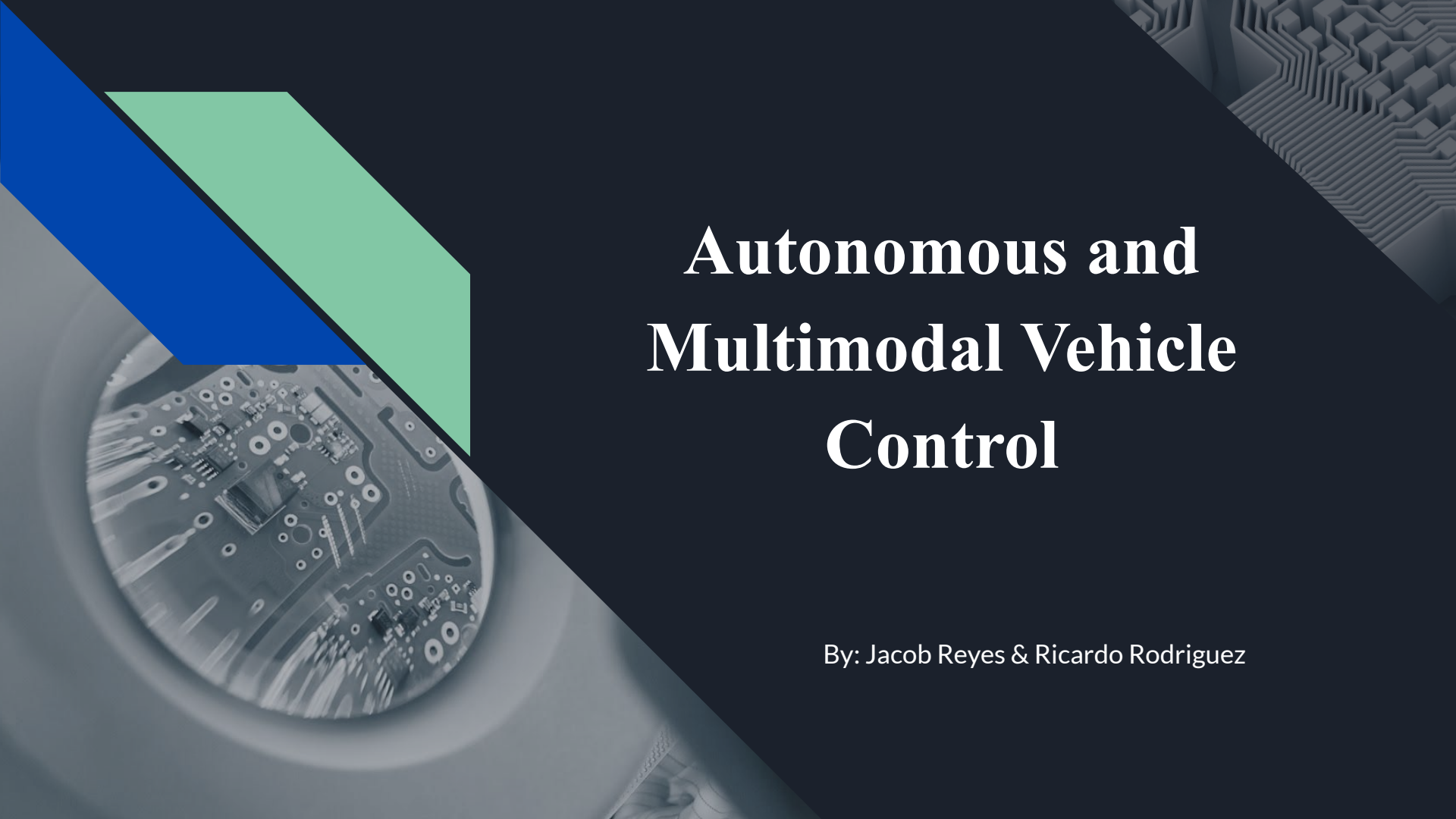


Autonomous and Multimodal Vehicle Control

By: Jacob Reyes & Ricardo Rodriguez

Project objective / Goals

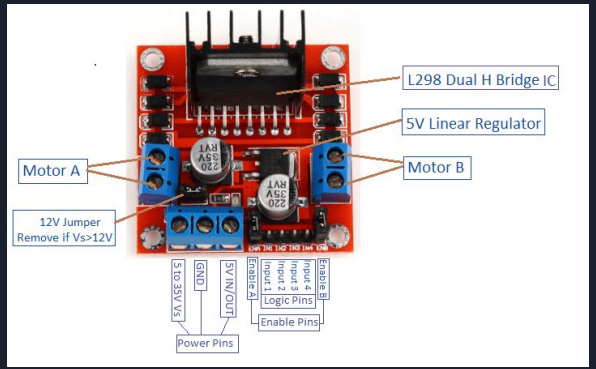
The main objective of this project is to design and develop a vehicle with two modes of operation, capable of switching between them wirelessly.

A wireless device will be used to interface with the vehicle, enabling the controller and configuring its operating mode.

The robotic vehicle operates in two modes: Following Mode and Remote Control Mode

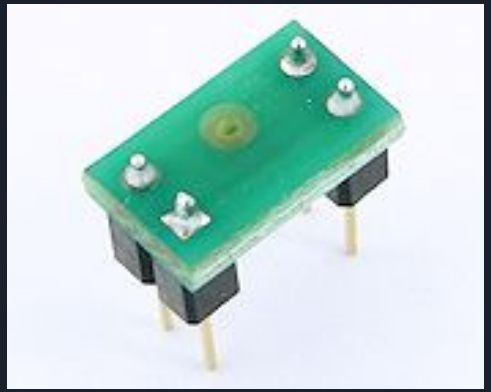
Description of System Design - Hardware

- TI MSP-EXP432P4111 - Microcontroller
- HC-SR04 - Ultrasonic Sensor
- L298N H-Bridge - Motor Driver
- 12-V Battery Pack equipped with an On/Off switch
- T Star DSTR - Robot Chassis

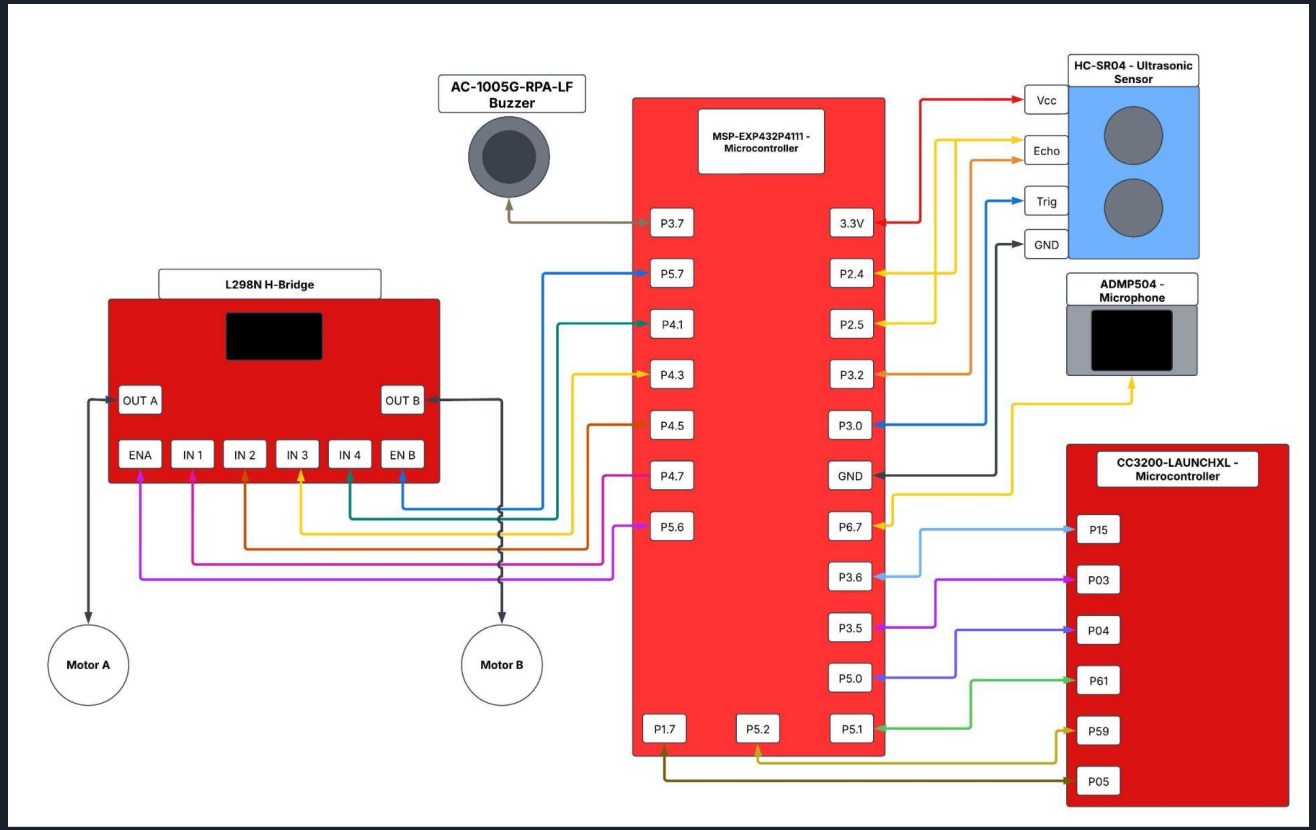


Description of System Design - Hardware

- CC3200-LAUNCHXL - Microcontroller
- AC-1005G-RPA-LF - Buzzer
- ADMP504 - Microphone

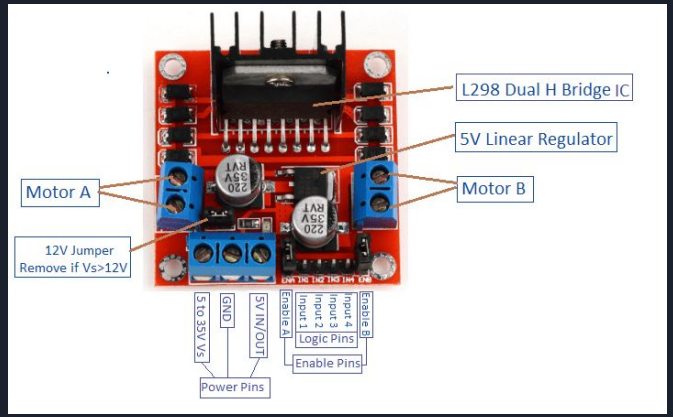


All Connections to the MSP432



L298N H-Bridge Motor Driver Input Table

Input						Output	
Enable A	Input 1	Input 2	Input 3	Input 4	Enable B	Motor A (1&2)	Motor B (3&4)
	0	0	0	0		0	0
PWM Signal	0	1	0	1	PWM Signal	1(Reverse)	1(Reverse)
PWM Signal	1	0	1	0	PWM Signal	1(Forward)	1(Forward)
	1	1	1	1		0	0
PWM Signal	1	0	0	1	PWM Signal	Rotate Right	
PWM Signal	0	1	1	0	PWM Signal	Rotate Left	

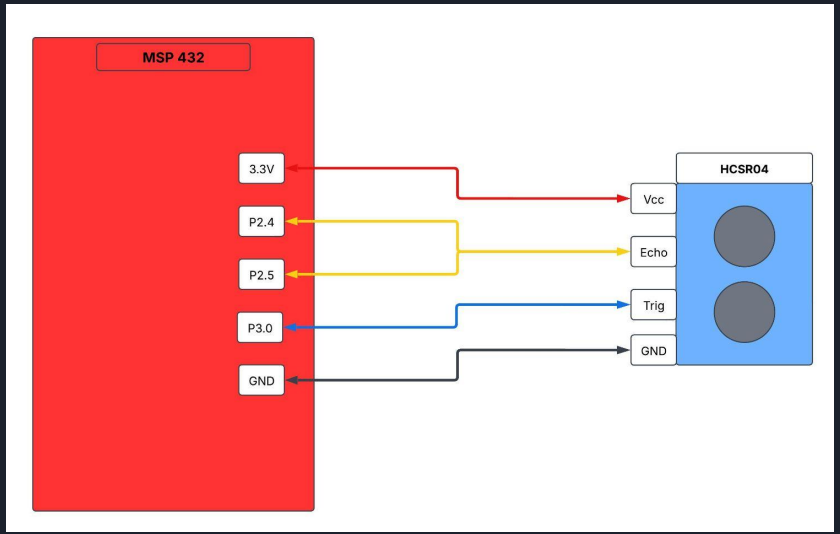


Ultrasonic Sensor Circuit & Distance Calculation

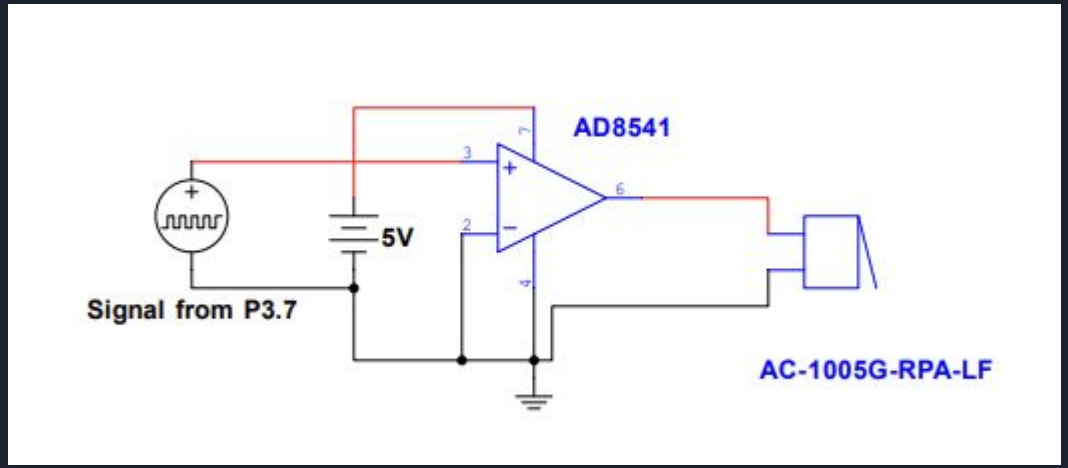
$$Distance = \frac{1}{2} * 343(m/s) * \frac{1}{3MHz} * (\alpha)$$

Sensor Equation Used:

- $\propto$ = Decimal Value of Value Generated by Sensor
- Final Units: Meters



AC-1005G-RPA-LF Buzzer Circuit / Frequency Calculation

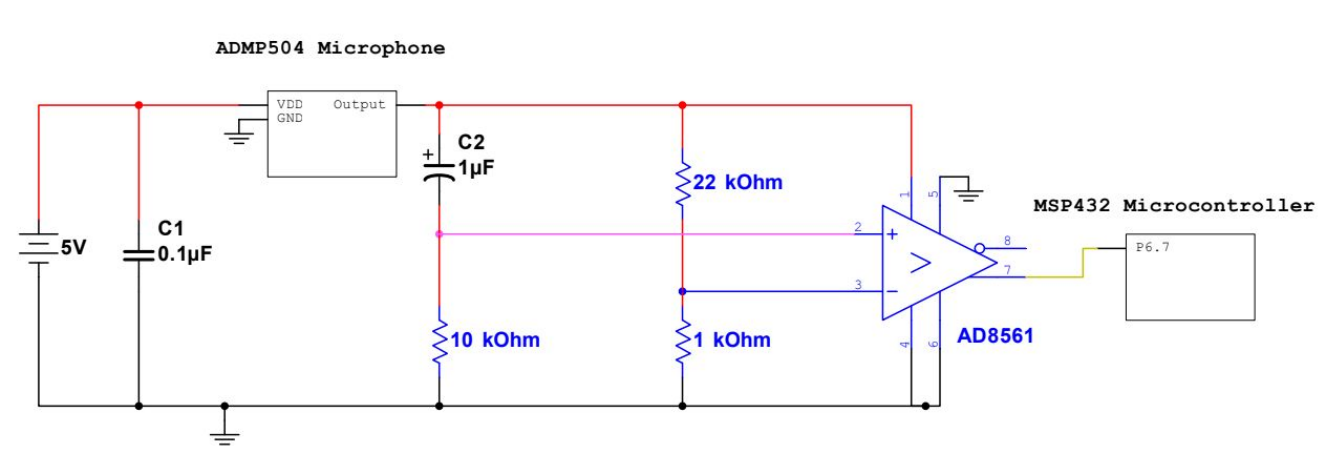


Output Frequency Calculation:

$$f_{OUT} = \left(\frac{3MHz}{f_c}\right) - 1$$

Where: f_c = Desired Frequency

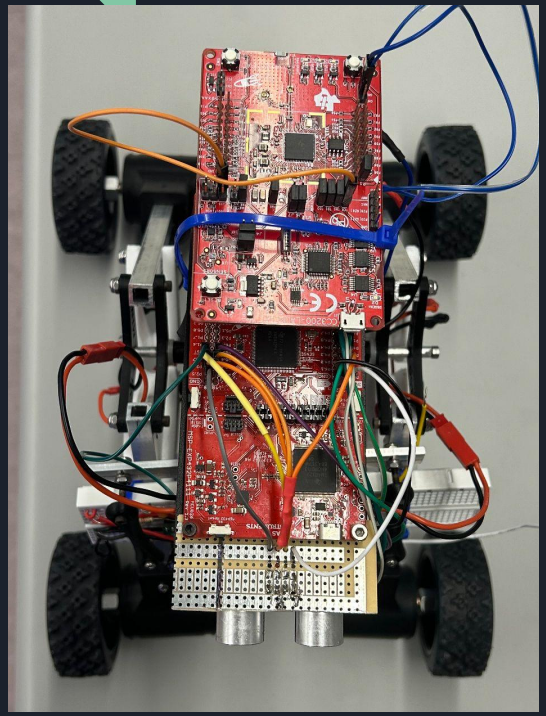
ADMP504 Microphone Circuit

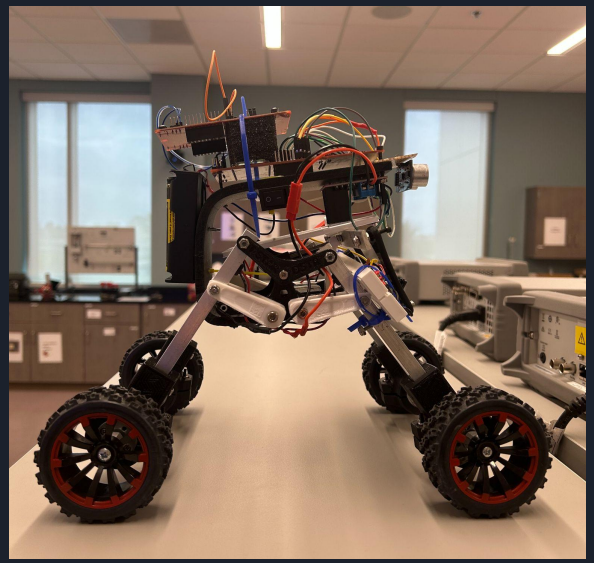


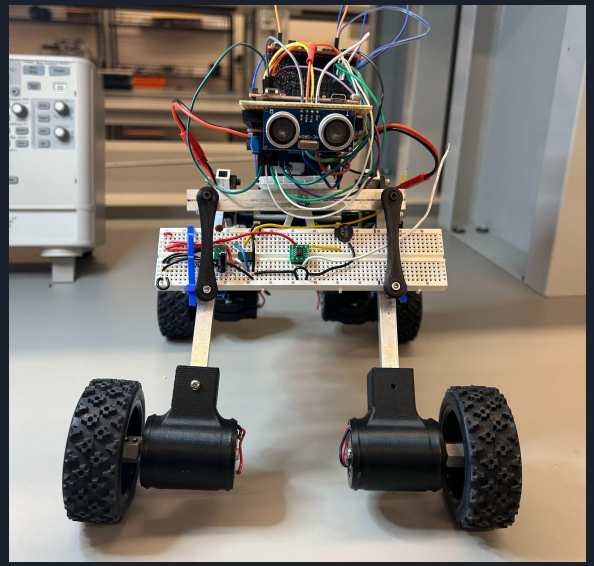
Vref = 0.2V
Typical Mic Output = 0.3V

$$V_{Ref} = \left(\frac{R_2}{R_1 + R_2}\right) V_{in}$$

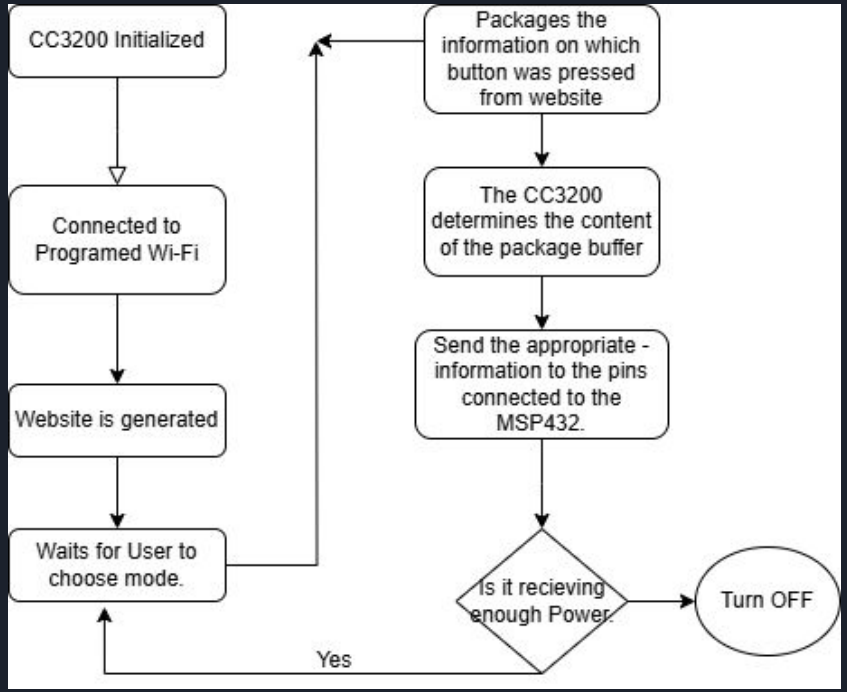
Completed Vehicle







Overall Schematics Done -Software Portion (CC3200 LAUNCHXL Board)



Portion of Code Alignment -Software Portion (CC3200 LAUNCHXL Board)

```
// if the current line is blank, you got two newline characters in a row.
// that's the end of the client HTTP request, so send a response:
if (strlen(buffer) == 0 ||Page == 0) {
  // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
  // and a content-type so the client knows what's coming, then a blank line:
  client.println("HTTP/1.1 200 OK");
  client.println("Content-type:text/html");
  client.println();
  client.println("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">");
  // the content of the HTTP response follows the header:
  client.println("<html><head><title>MSP CAR - Directinal and Activation</title></head><body align=center>");
  client.println("<h1 align=center><font color=\"red\">ESET 369 FINAL</font></h1>");
  client.print("MODE <button onclick=\"location.href='/Following'\"><font color=\"green\">Following</button></font>");//Mode
  client.println(" <button onclick=\"location.href='/Control'\"><font color=\"red\">Control</button><br></font>");//Mode
  client.print("Turn OFF! <button onclick=\"location.href='/TURN_OFF'\"><font color=\"red\">TURN OFF</button></font>");//TURNED_OFF
```

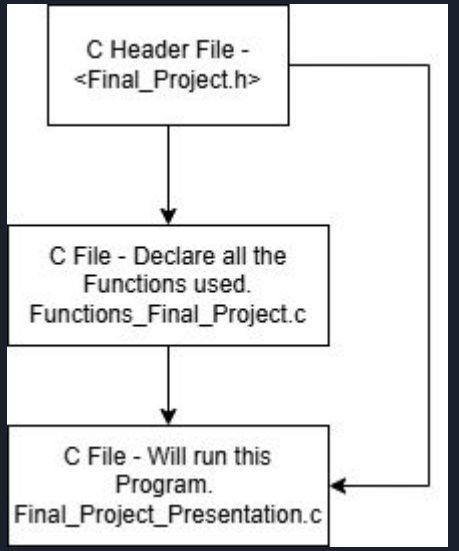
```
if (endsWith(buffer, "GET /TURN_OFF")) {
  digitalWrite(RED_LED, LOW);                    // TURN OFF THE ROBOT
  Serial.print("\nOFF!\n");
  digitalWrite(Power, LOW);
  Page = -1;
}
```

Overall Schematics Done -Software Portion (TI-MSP432 Board)

The <Final_Project.h> works as a C Header File that will declare all the functions used as well as all the variables used in an all the C files.

The <Functions_Final_Project.c> will store all the functions that will be used in our project, this file will not have a main function.

The <Final_Project_Presentation.c> will have the code with the main function. This will make the MSP432 board run this program and call upon the desired functions from the secondary C File that was also called upon on the Header File.



C Header File - <Final_Project.h>

The image to the right is the complete header file we are creating. It contains various functions that just initialize the required pins for a specific action.

Interesting Fact:

To call a variable from another C file we will have to initialize the variable in the header file with the following code.

```
extern int Clap;
extern int Enable_WiFi;
```

```
#ifndef Final_Project
#define Final Project

//Functions
void sensor_activation(void);
void delayMs(int n);
float conv(int x,int y);
void Sonic_Init(void);
void Motor_Init(void);
void state(float sensor_dis);
void motors(float, float, int, int);
void UARTInit(void);
void TX(char text[]);
int RX(void);
void TX_Float(float temp, int range);
void WiFi_init(void);
void mic_init(void);
void left_right(void);
void stationary(void);
//Following
void Following(void);
//Control
void WiFi_Control(void);
void buzzer_init(void);
void map(void);
void foreward_sound (void);
void reverse_sound(void);
//Variables needed in both Files
extern int Clap;
extern int Enable_WiFi;

#endif
```

Secondary C File - <Functions_Final_Project.c>

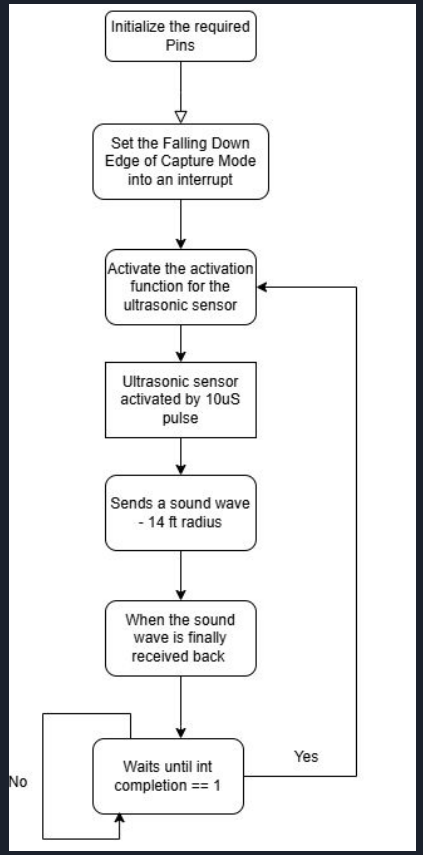
In this secondary C File specialized in storing all the functions we have created 21 separate functions. Some of these include:

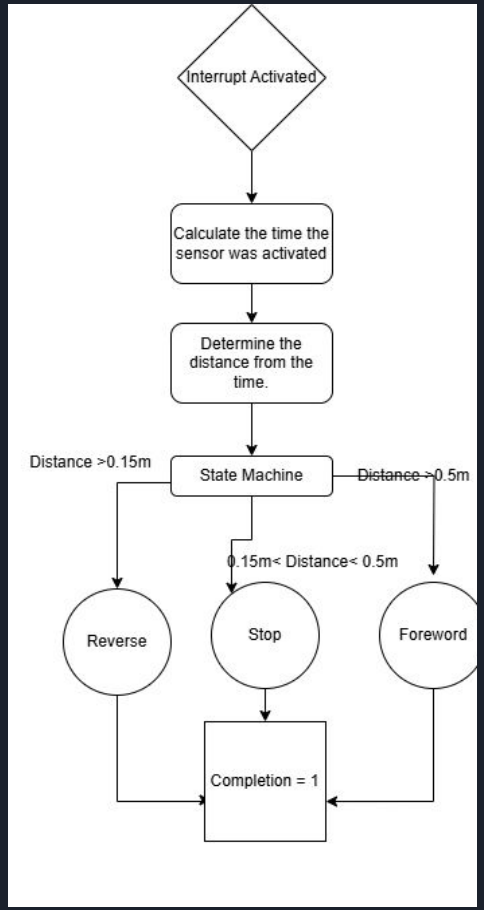
- 6 initialized Functions
- 2 Calculation Functions
- 1 Mapping Function
- 1 Timer Function
- 2 Interrupt Function
- 2 Main Function
    - Wi-Fi Control - Function
    - Following - Interrupt/Function Hybrid

Following - Schematic Function

This is the basic schematic for the Ultrasonic Sensor utilizing a real time event event as well as implementing functions.

In the next slide, we can see the programing for the state machine.



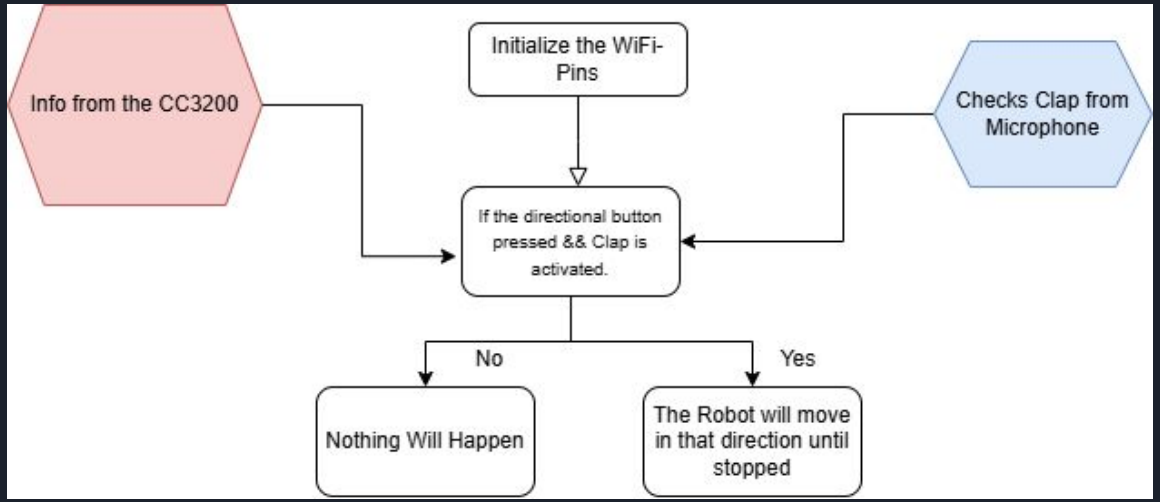


State Machine Function for Following - Example

```
void state(float sensor_dis){           //This function decides how the robot should move
        float far, mid_1, small;                        // with the input of the sensor disctance.
        far = 1.1;
        mid_1 =0.5;
        small = 0.25;

        if (sensor_dis >= far){ //If person out of 1.1 meter range
                motors(0.0, 0.0, 1, 1);                 //Robot will not move
                TX("    Steady");
                return;
        }
        else if(( sensor_dis > mid_1) && (sensor_dis < far)){
                motors(0.3, 0.3, 1, 1);         //Person in between 1.1>x>0.5 meter range
                TX("    Foreward");                                     //Robot will accelerate with a 50% PWM
                return;
        }
        else if(( sensor_dis <= mid_1) && (sensor_dis > small)){
                motors(0.0, 0.0, 1, 1);                 //If Person between 0.25 < x <= 0.5 meter range
                TX("    Steady");                                                       //Robot will stop
                return;
        }
        else if(sensor_dis <= small){   //If person closer than 0.25 meters
                motors(0.3, 0.3, 0, 0);                 //Robot will reverse back until in 0.25 range.
                TX("    Reverse");
                return;
        }
        return;
}
```

WiFi Control Function

This function allows us to determine what type of direction we want from the MSP Robot. The next slide will contain the actual function where the pins are being checked for each direction.



Wi-Fi Control Function Program.

```
void WiFi_Control(void){
        while(((P3 -> IN & 0x20)== 0)){
                //Foreward
                if(((P5 -> IN & 0x01)!= 0)){
                        motors(0.9,0.9,1,1);
                }

                //Reverse
                else if(((P5 -> IN & 0x02)!= 0)){
                        motors(0.9,0.9,0,0);
                }

                //Left Shifting
                else if(((P5 -> IN & 0x04)!= 0)){
                        motors(0.99,0.99,0,1);
                }
                //Right Shifting
                else if(((P1 -> IN & 0x80)!= 0)){
                        motors(0.99,0.99,1,0);
                }
                if(((P1 -> IN & 0x80)== 0)&&((P5 -> IN & 0x04)== 0)&&((P5 -> IN & 0x02)== 0)&& ((P5 -> IN & 0x01)== 0))
                        motors(0,0,1,1);
        }
}
```

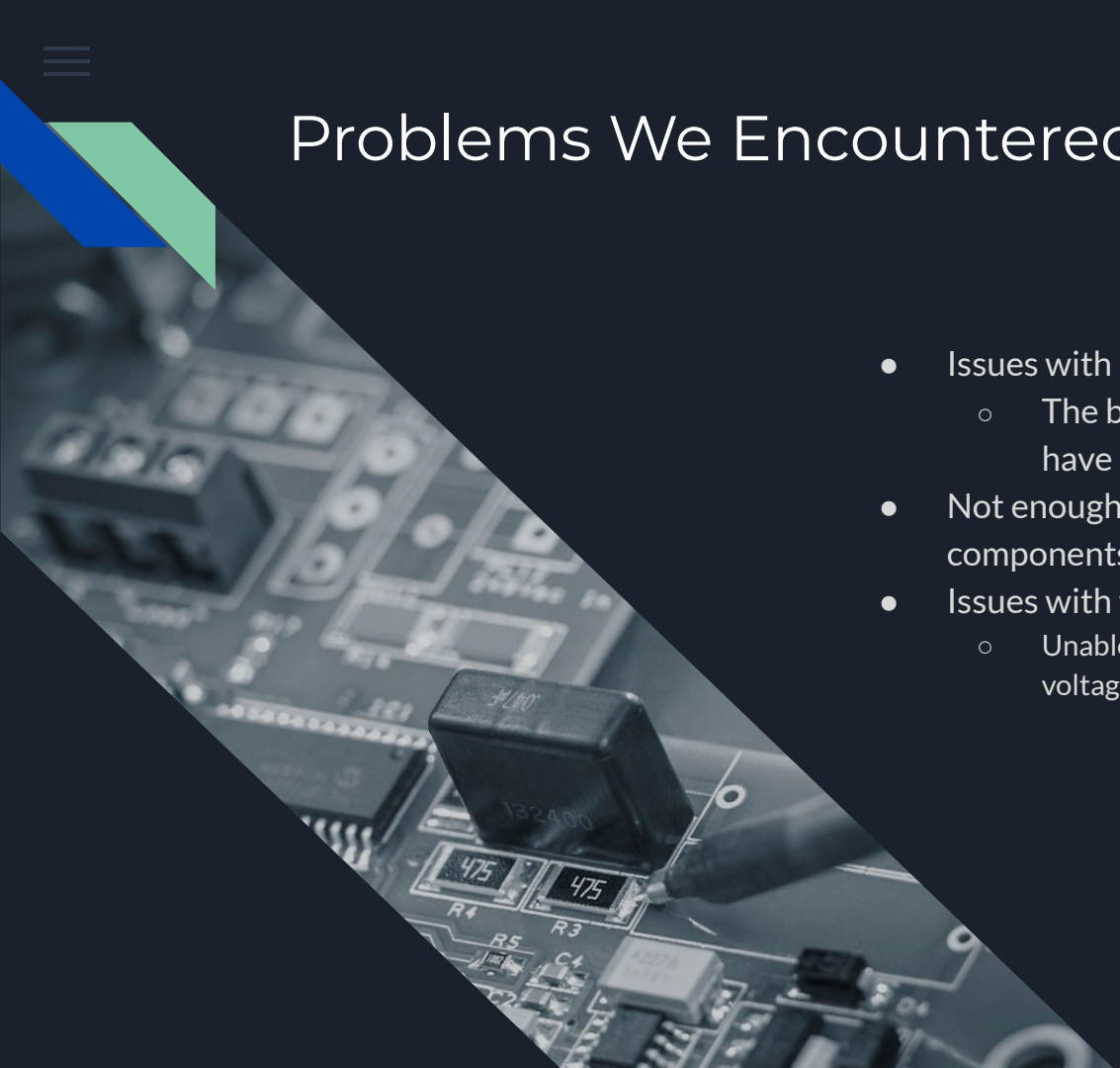
Primary C - File Programed

This File will have the most minimalist code to allow readability for the user as well as following popular programing conventions.

```
#include "msp.h"
#include <stdio.h>
#include <stdlib.h>
#include "Final_Project.h"

int main(void){
        UARTInit();
        Sonic_Init();           //Initializes the Sensor
        Motor_Init();           //Initializes the Motors
        mic_init();
        buzzer_init();
        map();
        WiFi_init();

        while(1){
                if(Clap == 1){
                //      stationary();
                        while(((P3 -> IN & 0x20)!= 0)){
                                Following();
                        }
                        while(((P3 -> IN & 0x20)== 0)){
                                Enable_WiFi = 1;
                                WiFi_Control();
                        }
                }
                else if(Clap ==0){
                        TX("Here!");
                //      stationary();
                        //WiFi_Control();
                }
        }

        return 0;
```

Vehicle Demonstration

Problems We Encountered.



- Issues with battery supply
  - The batteries died fairly quickly and do not have enough current to rotate the vehicle.
- Not enough space on the chassis for mounting components.
- Issues with the microphone for clap detection
  - Unable to use an Op-Amp to amplify the output voltage of the microphone

Future Improvements

Features that we would like to add / improve upon for future modifications would include:

- Adding a high power battery pack
- Improving the user interface on the vehicle website
- Adding an Op-Amp that will amplify the output voltage from the microphone
- Standardizing the wire colors on the entire vehicle
- Constructing a shell to enhance the appearance of the vehicle

Thank you!

Any Questions?