Internet Controlled Smart Movable Platform
Design Review

Team 46

Kecheng Liu
Yubo Liu
Yigao Shao
TA: Mustafa Mukadam

University of Illinois at Urbana-Champaign
March 3, 2013

Table of Content

1.0 Introduction

1.1 Statement of Purpose

The goal of this project is to build a real-time, WIFI controlled virtual-tour vehicle that enables users to "visit" any location with WIFI connection. Many network-controlled vehicles exist on the market, varying from hobbyist models to commercial products such as the VGo from Verizon Wireless. However, none has a complete system that includes a robust backend (software) and a versatile front-end (hardware). These include automatic driving, path memorization, environment/obstacle sensor, self-correcting/repairing network, and a functional website. The product we plan to develop is unique and will help revolutionize how people "visit" and view the world!

1.2 Objectives

In this project, we intend to create a complete virtual tour system that combines a robust network between our website and the moving vehicle (MV). In the backend, we will use a web server and a server on the MV to create a P2P connection between the user and the MV. Different users can safely control the MV from the users' computer. The user needs to create an account on our website, after which the user will be able to log in and take control of any idle MV. On the hardware side, we plan to implement various features that allow the MV to be easily controlled and autonomous. Considering the safety issue of the MV, we will implement an environment detector by using ultrasonic sensors. This allows our MV to automatically change speed and direction in emergencies such as seeing an obstacle. For friendly usage and control, a GPS will be installed on the MV to allow automatic controls and navigation.

1.2.1 Functions

The functions of our projects are listed below

1. Travels in different directions and speed
2. A live video camera to provide vision
3. A website for users to log in, choose the desired car which is provided by a car owner and control
4. Collision avoidance system to protect our car from unpredictable hazards
5. Four driving modes
   a. Full real-time manual control
   b. Programmed route control
   c. Automatic navigation via path memory to starting location
   d. Automatic control under GPS navigation.

1.2.2 Benefits

1. Access from long distance from any computers with internet connection
2. Robust internet connection
3. Robust software that is compatible with various network interface
4. Can have real-time tours through on-board live video camera
5. The MV has collision detect and avoidance system to protect the vehicle

2.0 Design

2.1 Block Diagram

Fig. 1 Overall Block Diagram

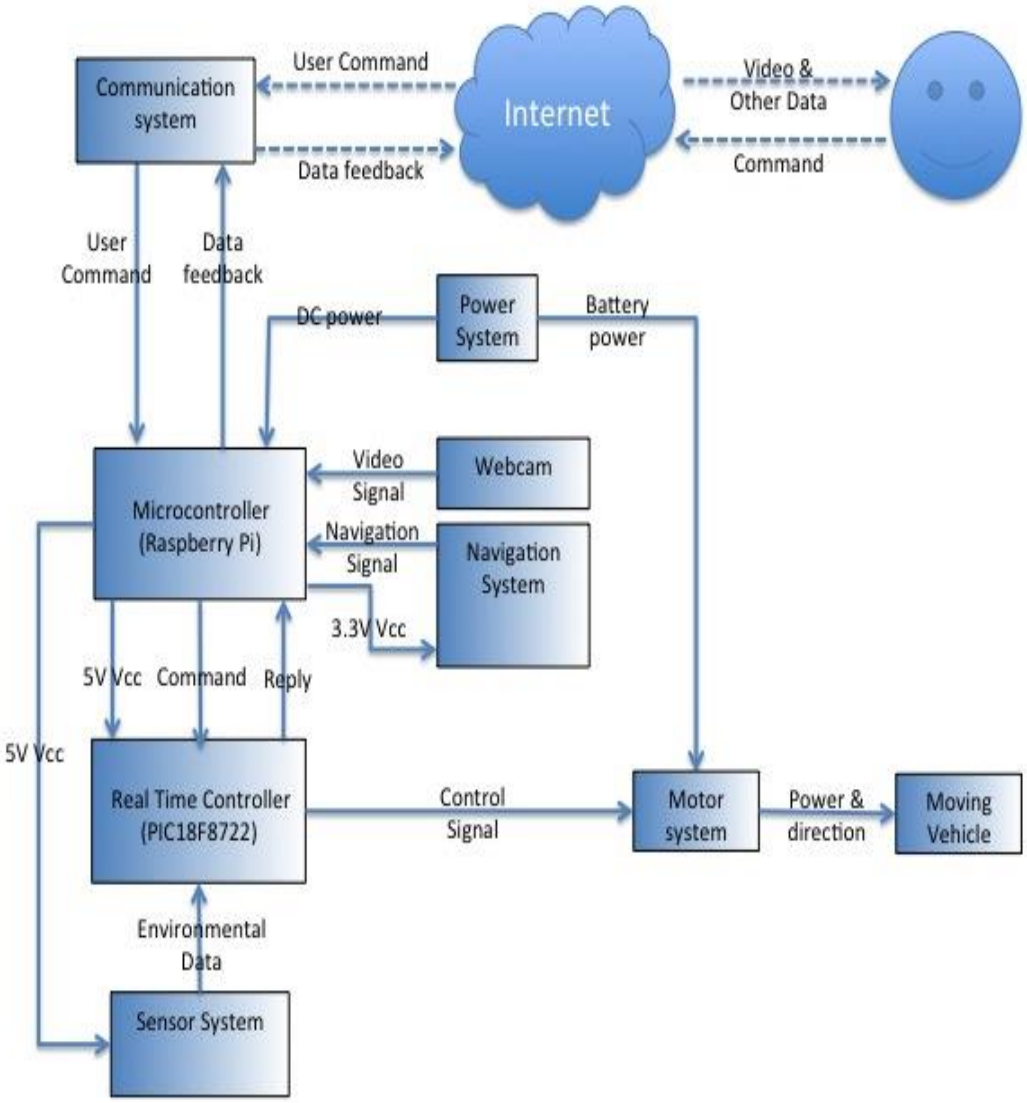


Fig 2. Middle Layer Layout

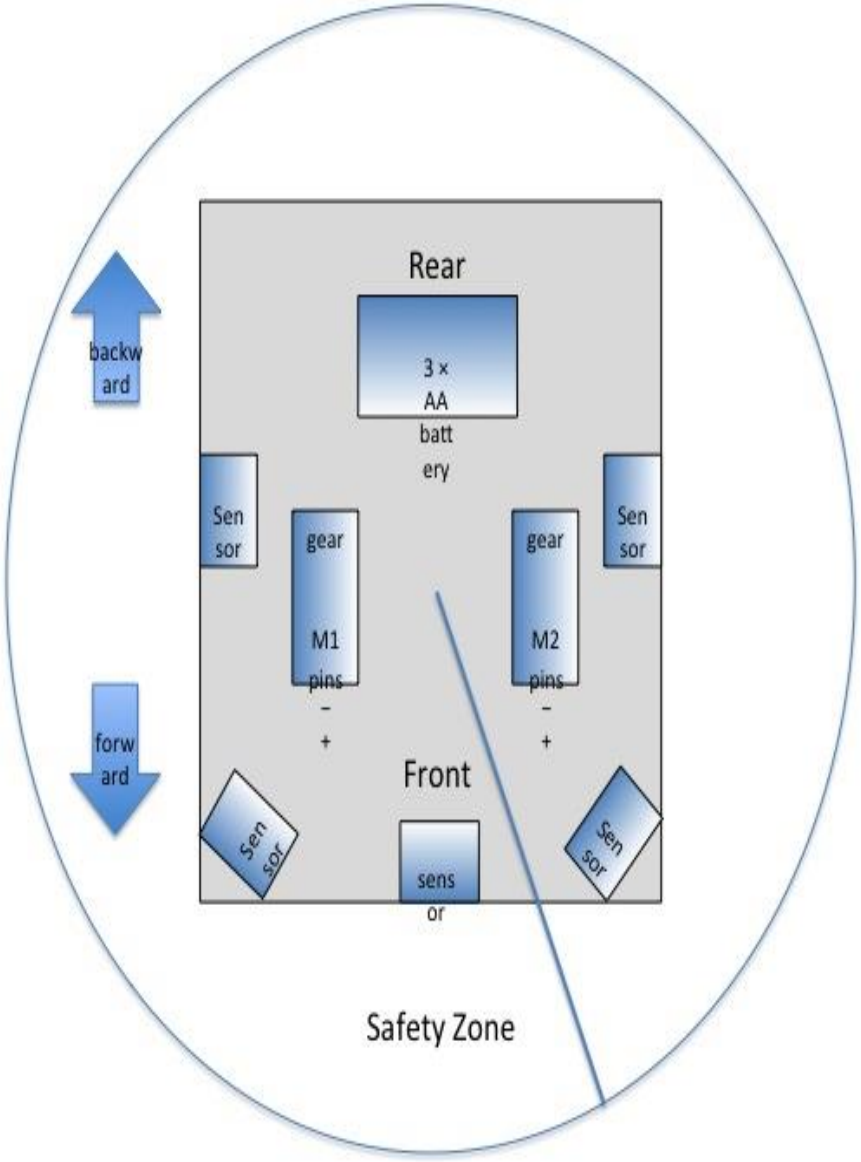


Fig 3. Bottom Layer Layout

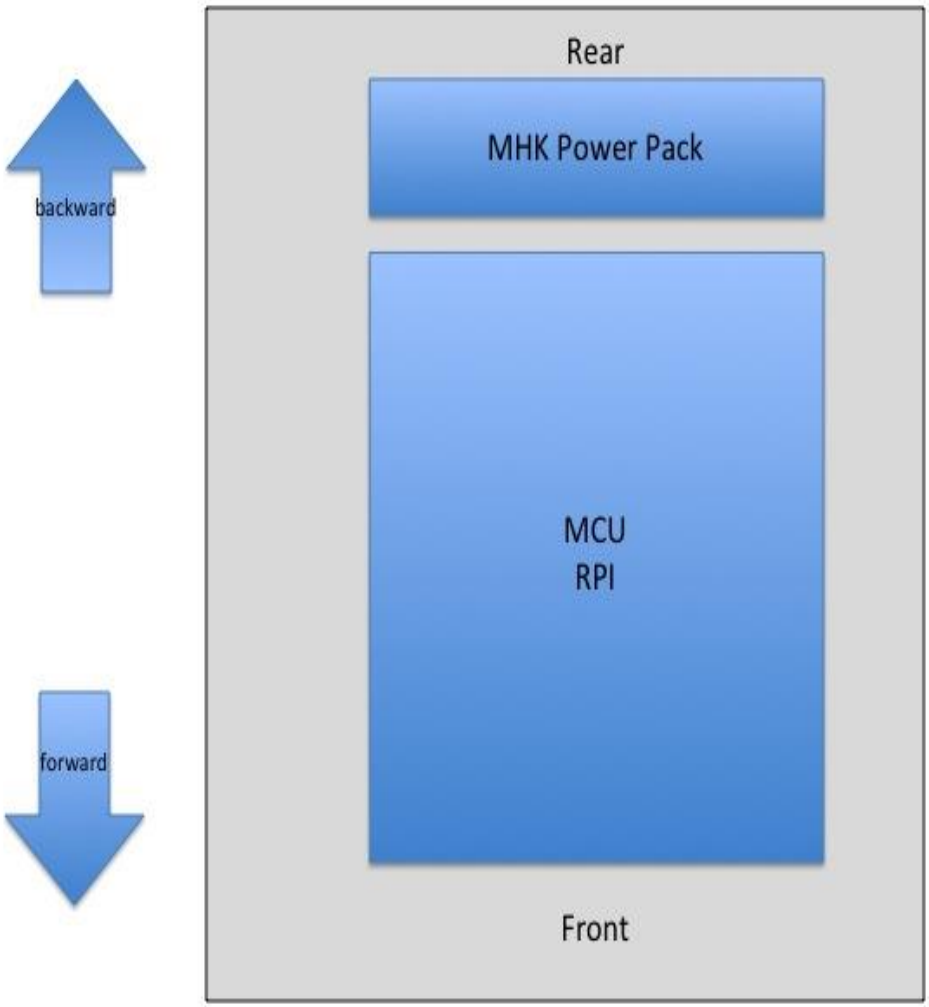
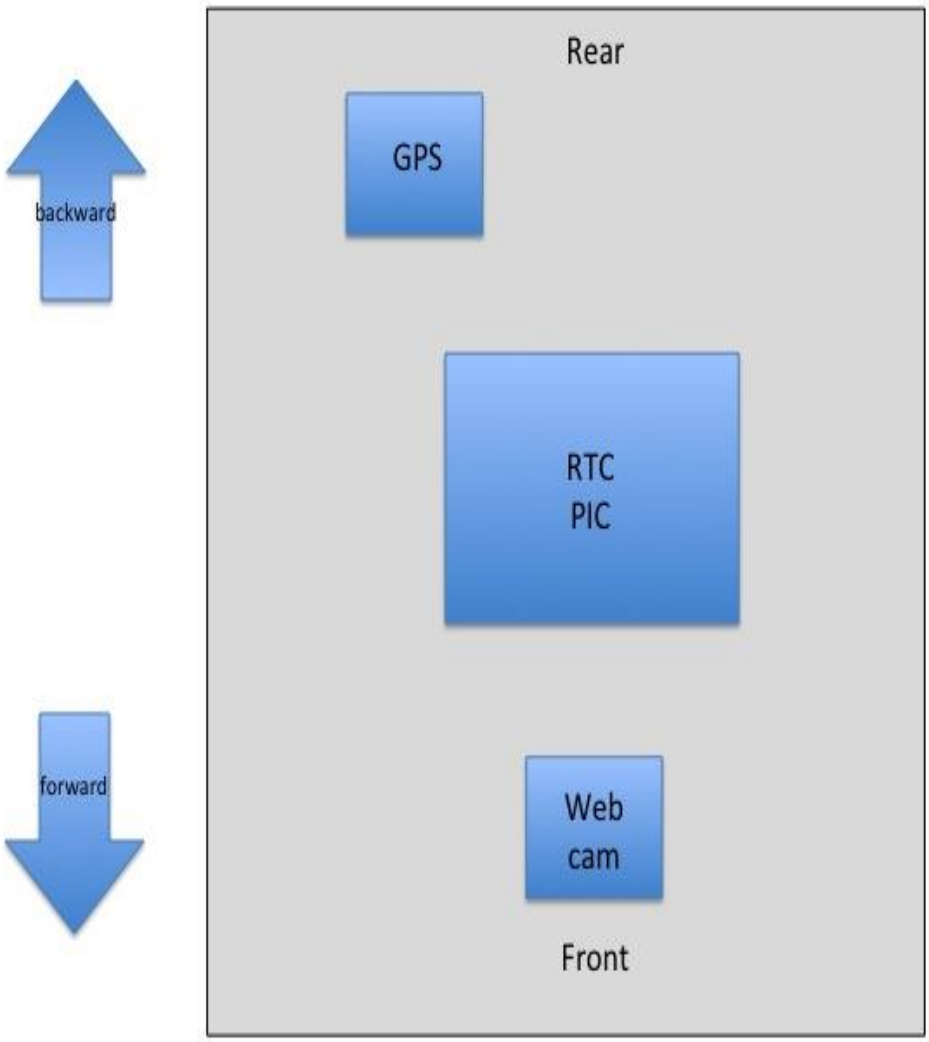


Fig 4. Top Layer Layout



2.2 Block Description

2.2.1 Hardware Blocks

Summary of Hardware Blocks

- Power Module
- Microcontroller Unit (MCU)
- Moving Vehicle (MV)
- Motor System Module
- Sensor Module
- Navigation System
- Real Time Controller (RTC)
- Video Module
- Communication System

Power Module

Input: The inputs to the power module are batteries. Since we have two separate power system modules, we have two kinds of batteries. The first is 3 AA batteries (4.5 V), and the second is the MHK 2200 mA-H Rechargeable Power Pack at 5V DC.

Output: The power module outputs power to the MCU (Raspberry Pi) at 5 V, the Motor System Module at 4.5 V. In addition, the MCU can output power to its connected subcomponents as seen in Fig 5.

Description: The Power Module's purpose is to deliver power to other electronic modules. The Power Module consists of two power sources to provide power to the different modules on the vehicle. The MHK Power Pack battery will be rechargeable, while the AA batteries can be easily replaced. Since the MHK Power Pack battery is more reliable and can last longer, and since the MCU can output power to other units, we prefer to use the MCU to provide power to other modules besides the Motor System Module. The Motor System Module needs a separate power system because it requires high current input (at least 250mA per motor). Fig 5 shows the power distribution of Power System and Table 1 shows the Power Budget Table for our project. It is calculated based on the total energy in the battery supplies and finding the peak power of each component via data sheet and testing. If the peak power of each component adds up to be less than that of the supply, then we are under budget.

Fig 5. Power System

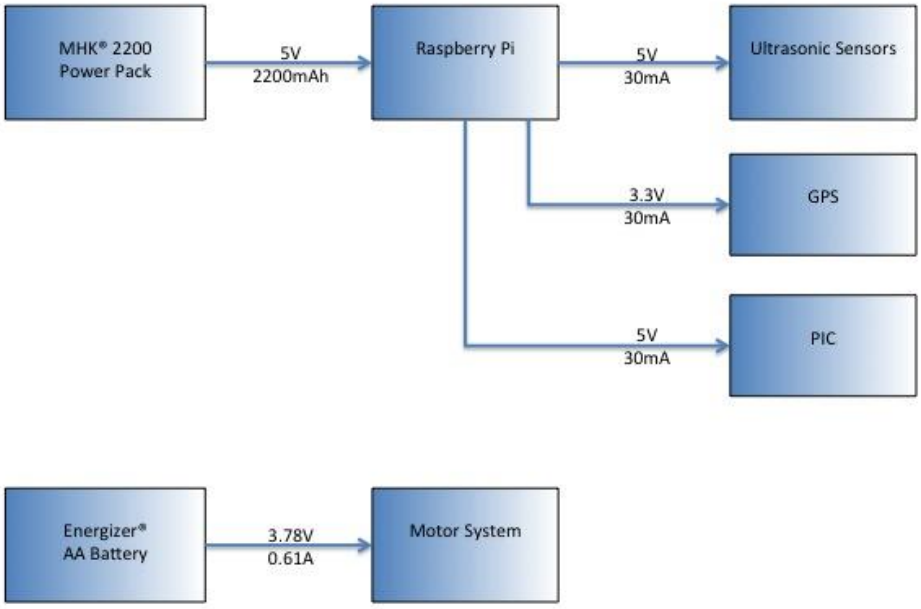


Table 1. Power budge table

Energy available		Energy consumption	
Type	$Quantity \times rated\ voltage\ (V)$ $\times rated\ electrical\ charge\ (Ah)$	Type	$Quantity$ $\times rated\ power\ (W)$ $\times 1\ time(h)$
AA battery	$3 \times 1.5\ \times 2.2 = 9.9\ Wh$	Motor (full load)	$2 \times 3.375W \times 1 = 6.75Wh$
		Total	$6.75\ Wh$
MHK power pack	$1 \times 5 \times 2.2 = 11\ Wh$	RPI	$1 \times 3.5 \times 1 = 3.5\ Wh$
		Sensors	$5 \times 15\ mW \times 1 = 75\ mWh$
		GPS	$1 \times 145.2\ mW \times 1 = 145.2\ mWh$
		PIC	$1 \times 10\ mW \times 1 = 10\ mWh$
		Total	$3.73\ Wh$
Grand Total	$11 + 9.9 = 20.9\ Wh$	$10.48\ Wh$	

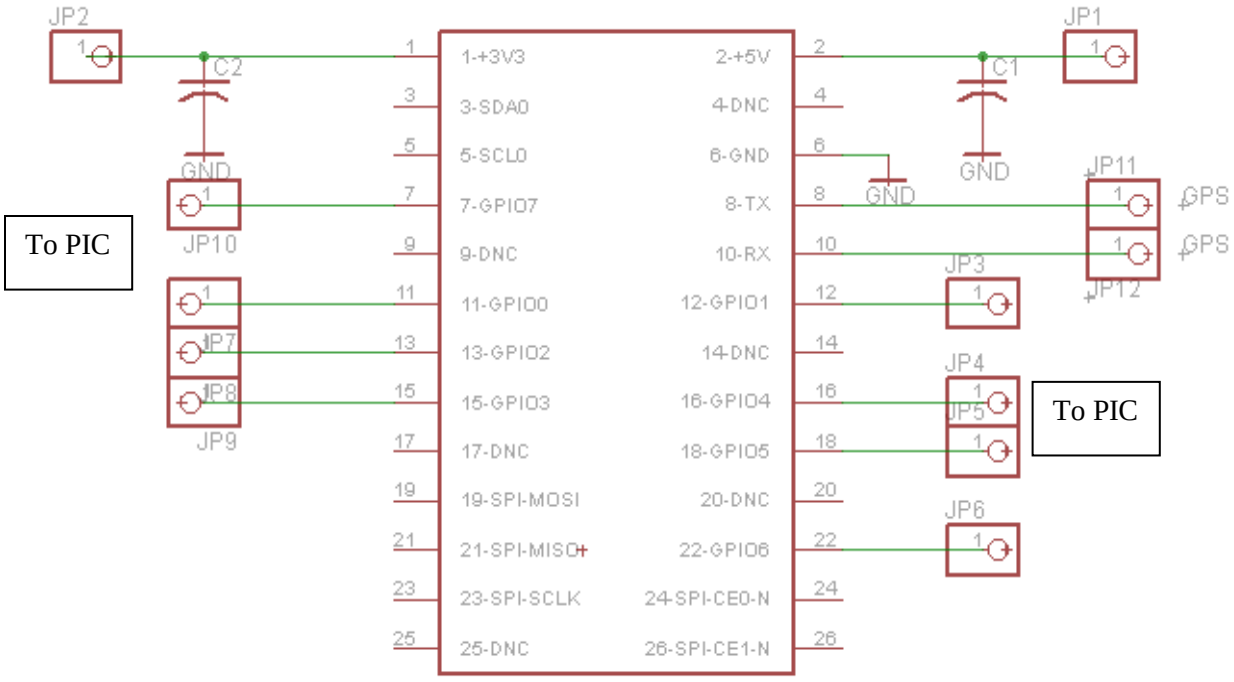
Microcontroller Unit (MCU)

Input: The MCU's input is from the webserver on user's side, the Real Time Controller (RTC), the Power Module, the Navigation System, and Video Module. The signals include user's command, sensors analog output, power, GPS data, and video data. All these data are processed to make the next decision for the system.

Output: Since the Raspberry Pi can output power supply, we use the MCU to provide power for the RTC, Sensor Module, and the Navigation System. The MCU also outputs the system's current data such as video and location back to the user's webserver. In addition, the MCU sends control signals to the RTC so the RTC can set and send the corresponding navigation signals to the Motor System.

Description: The MCU is the heart of our system. It consists of Raspberry Pi, which is a powerful microcontroller with a reasonable price tag. The MCU handles data from the user's webserver as well as data from the MV's side. With the data communication going both ways, the MCU will be able to produce the signals for the users to see on the website and signals for the navigation of the MV. Fig 6 below shows a schematic of connections on the MCU

Fig. 6 Raspberry Pi Connections



Moving Vehicle (MV)

Input: The input to the MV is from the Motor System Module. The Motor System Module allows the MV to move forward (F), back (B), left (L), and right (R).

Output: In response to the input of the Motor System Module, the MV will move in different direction

Description: The MV is the platform in which our hardware is installed upon. It is a toy car that was purchased from Mr. Basic and assembled by us. The movement and direction are controlled by users or by automation. The max speed of the MV on bare floor is found to be 38.8 cm/s (without external modules).

Motor System Module

Input: The input of the Motor System Module is power from the Power Module. Also, the Motor System Module receives input from the Real Time Controller (RTC). The RTC sends control signals to the motor system module to instruct each of the motor's movement.

Output: The Motor System Module outputs torque to the gears to move the wheels. This allows the MV to move at a maximum rate of 38.8 cm/s as stated in the MV section.

Description: The Motor System Module consists of two sections: two DC motors and an H-Bridge. Together, H-Bridge and the DC motors are responsible for the physical movement of the MV. The H-Bridge is connected to the motor to control the movement of the motor. Depending on the signal the H-Bridge receives from the PWM, the H-Bridge tells which motor to move and the direction the motor moves. The DC motors are the other part of the Motor System. The motors are used to provide the movement of the MV's wheels depending on the signals from H-Bridge. Please refer to Motor System Theories section below for detailed analysis.

Motor System Theories

The layout of the MV is shown in Fig 2. Two motors are placed on each side of the MV: Motor 1 (M1) is on the left side and Motor 2 (M2) is on the right side.
The MV moves forward if M1 spins counter clockwise (CCW) and M2 spins clockwise (CW), viewed from the front side of the MV. For backward motion, we simply flip the direction of motor spinning. The MV turns to the left if M2 spins CW and M1 remains stationary; it turns to the right if M1 spins CCW and M2 remains stationary.

The spinning direction of both motors is determined by the direction of current flow. M1 spins CCW (CW) and M2 spins CW (CCW) if current flows from their positive (negative) pin to negative (positive) pin and the MV moves forward (backward). Based on this setup, we can determine how M1 and M2 are connected with H-bridges.

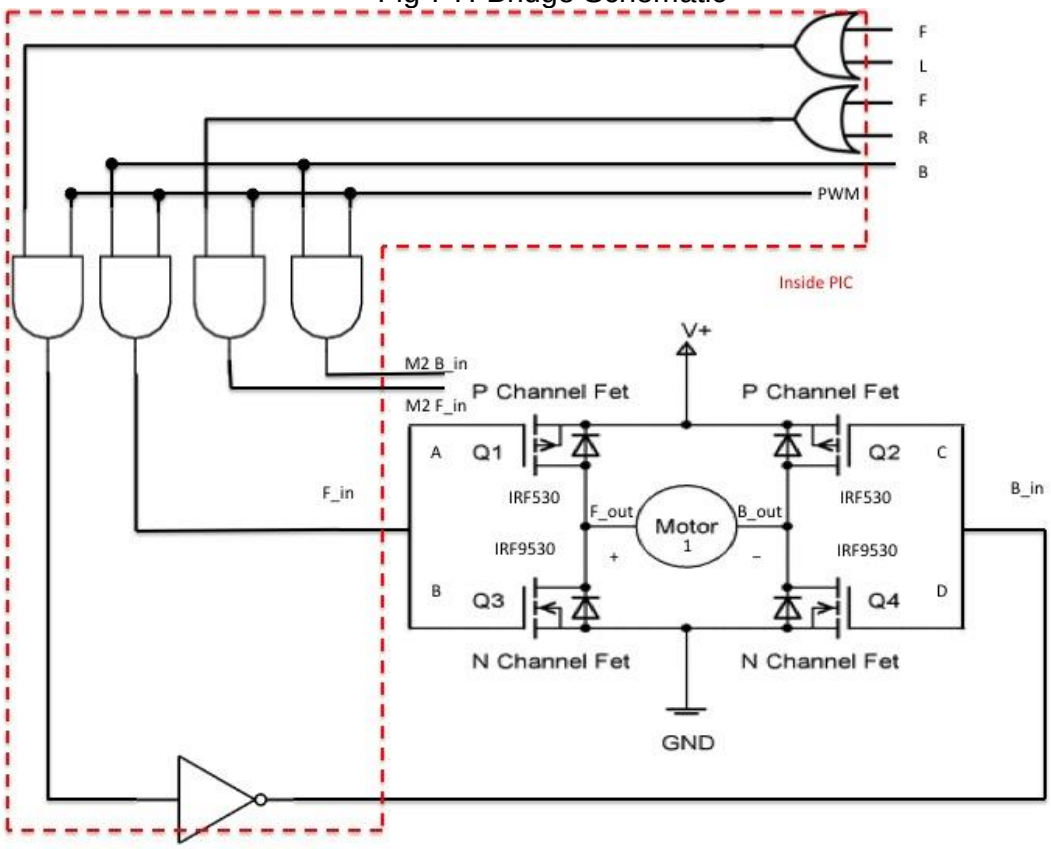
The schematic of H-bridge is shown in Fig 7. H-bridge consists two n-type MOSFETs and two p-type MOSFETs The gate of each MOSFET acts as the input of the H-bridge and noted as A, B, C and D. Based on the fact that for n-type and p-type MOSFET, current will flow across the drain and source when gate voltage is respectively above and below

threshold voltage (3.3V in our case), we can make up a truth table, as shown in Table 2, for the digital relationship between gate voltage level and MV moving directions.

Table 2 Truth Table for H-bridge Digital Control

A	B	C	D	MV Moving Directions
1	0	0	1	FWD
0	1	1	0	BWD
1	1	0	0	Break
0	0	1	1	Break

Fig 7 H-Bridge Schematic



Logic 1 and 0 represents voltage level above and below the gate threshold voltage, respectively. From the truth table we can acquire two Boolean expressions for MV motions as equation (1) and (2):

$$FWD = AB'C'D \quad (1)$$
$$BWD = A'BCD' \quad (2)$$

Because MV can only move when AD and BC are excited separately, we connect A and B, C and D together as two inputs of the H-bridge and noted as: forward input (F_in) and backward input (B_in). Two outputs of the H-bridge are: forward output (F_out) and backward output (B_out). F_in and B_in receive input signals from PIC; F_out and B_out are connected to the positive and negative pins of the motors. According to the discussion above, the MV will move forward when current flows from positive pin to negative pin for both M1 and M2; therefore, the positive pins for both M1 and M2 are connected to the F_out and the negative pins are connected to the B_out.

Below is the Control logic for the MV. Because MV will receive four commands from user: forward (F), backward (B), left turn (L) and right turn (R), and receive PWM signal from RTC, the truth table for MV can be constructed and shown in Table 3

Table 3 Truth Table for MV Digital Control Signal

F	B	L	R	PWM	M1	M2
1	0	0	0	1	FWD	FWD
0	1	0	0	1	BWD	BWD
0	0	1	0	1	FWD	Break
0	0	0	1	1	Break	FWD
X	X	X	X	0	Break	Break

From the truth table Boolean expressions can be determined as equation (3) to (6)

$$M1_{FWD} = (F + L)PWM \quad (3)$$
$$M1_{BWD} = B \times PWM \quad (4)$$
$$M2_{FWD} = (F + L)PWM \quad (5)$$
$$M2_{BWD} = B \times PWM \quad (6)$$

$M1_{FWD}, M1_{BWD}, M2_{FWD}, M2_{BWD}$ are four inputs for two H-bridges and will be connected to F_in and B_in separately.

Additionally, to prevent the hazard from backward EMF current due to motor spinning, we placed diodes in parallel with each MOSFET to protect our sources.

Sensor Module

Input: The Sensor Module's input are input data from five LV-MaxSonar EZ0 and EZ1 ultrasonic sensors mounted on the front, two sides of the MV, and the two front corners. In addition, it will take inputs from the MCU for 5VDC supply. These sensors will send out thirteen 42kHz waves, which will be detected by the receiver on the EZ0 module. Then the signal will be filtered and amplified and fed to the PIC16F676 (all pre-packaged on module) to process the signal.

Output: The analog (AN) output of the sensors will be used as the outputs of the module. The analog signal is outputted from the on-module PIC6F676 and goes through a buffer. This signal has a scaling factor (SF) of

$$Distance = \frac{Vcc}{512}\ per\ inch \quad (7)$$

So for example, at Vcc = 5V, the analog signal will yield approximately 9.8mV per inch. This signal is fed to the Real Time Controller Module to convert to digital signal that can be used for further processing of the data such as automatic navigation and avoiding obstacles.

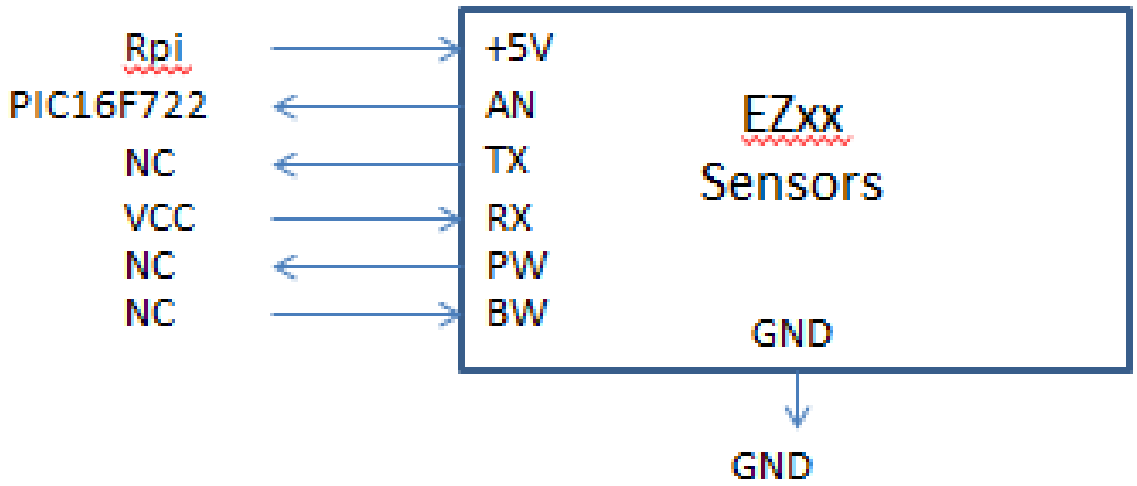
Description: Three sensors will be used in total to implement the Sensor Module. There will be one EZ0 mounted on the front of MV, and one EZ1 mounted on left and right side of MV, and one EZ1 mounted on each of the front corners. The EZ0 and EZ1 have similar specs; the main difference is in their sensitivity. EZ0 has better sensibilities and can see longer without interference. Thus, the front needs the most sensitivity to detect upcoming obstacles. The sides are supplement sensors to enable collision avoidance system. The section below features more detailed information.

Sensor Module Theories and Design

The EZ1000 family ultrasonic sensor block diagram is shown below. We pick EZ 1000 family ultrasonic sensors for two reasons: first, it is easier to use because it provides us an output voltage linearly proportional to distance; second, it is analog so we do not need to worry about the difficult serial digital data configuration and we can use this analog data by feeding it to PIC and use ADC to convert it to digital format.

For the obstacle avoidance function, we will setup a 10cm safety zone, which is slightly above the minimum detection range of the ultrasonic sensor. MV will stop if any sensor detects an object within safety zone. The RTC will decrease PWM duty cycle in coordinate with detection range. With speed control and safety zone setup, MV could be protected from potential hazard.

Fig. 8 Sensor Schematic



Navigation System

Input: The input is satellite from the GPS module (EM-408) receiver.

Output: The output is a serial output that indicates the longitude and latitude of the MV's location.

Description: The Navigation System allows the users to know the MV's current location. It can be used navigate automatically between two known locations. The Navigation System Theories and Design below will feature more detailed information.

Navigation System Design

The GPS module we are using is EM408. It will give us the global coordinate in latitude and longitude. In order to enable chip's navigation capability, we need to know the surface distance between two coordinates. This can be calculated by using the Haversine Formula as shown in equation 8 and 9 by assuming earth is a perfect sphere with radius = 6371.009m.

$$a = sin^2\left(\frac{\Delta\varphi}{2}\right) + cos(\varphi_1)cos(\varphi_2) * sin^2(\frac{\Delta\lambda}{2}) \quad (8)$$

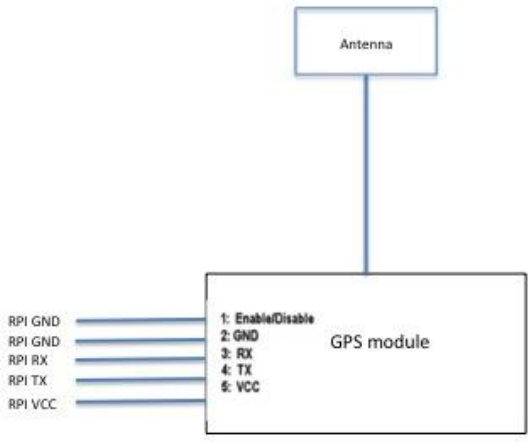
$$dis = 2 * r * sin^{-1}(\sqrt{a}) \quad (9)$$

Where φ is latitude and λ is longitude in degree and south and west will be considered as negative.

After we know the distance between two coordinates, we will need to find out the heading direction from one point to another. This can be done in two ways: to use a digital compass IC chip or to use algorithm. The algorithm to find the heading direction is to let the MV go to different direction and calculate the change of distance between temporary position and destination position; pick up the direction when distance is decreasing fastest and move along with it. We prefer to use a digital compass IC but we do not have extra serial I/O pins to configure this chip; therefore, we will use the algorithm. However, many problems and redundancy might occur due to the inaccuracy of GPS module.

To configure EM408 chip, we need to wire it up with MCU to pin Tx and Rx. They are serial I/O pins on RPI for receiving and sending data. The connection of our GPS system is shown in Fig 9:

Fig 9 GPS Schematic



Real Time Controller (RTC)

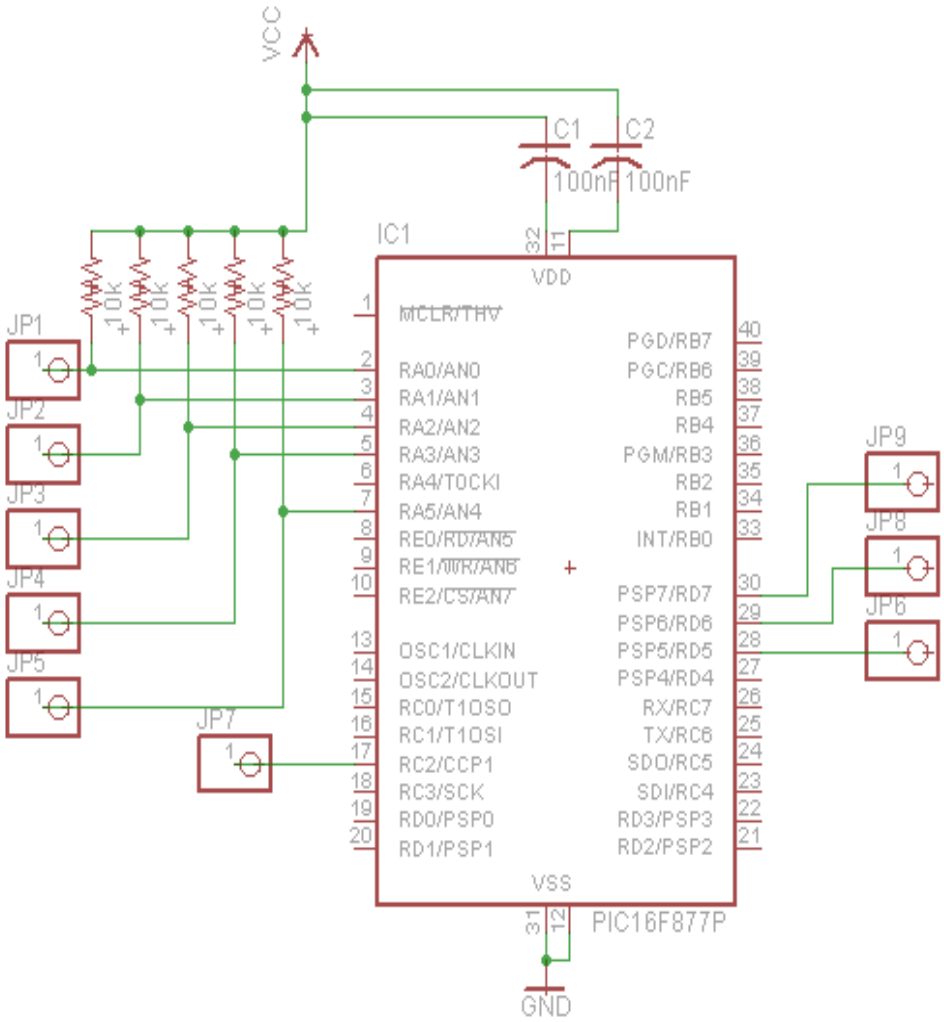
Input: The inputs include 5VDC supply power from MCU, analog signal from the Sensor Module, and control signal from MCU (master).

Output: the outputs include control signal to Motor Module System Module and feedback signal to the MCU as a slave. The output signals will tell the Motor System which motor to activate, what direction, and what speed. Lastly, the RTC relay any data from the Sensor Module back to the MCU.

Description: The RTC consists of the PIC16F877 and surrounding circuitry. The RTC's main purpose is twofold. First is to process the data from the Sensor Module to determine if there are any obstacles in the MV's path. The second is to provide the control signal for the Motor System Module based on the combination signal from the MCU and the Sensor Module. In doing this, the RTC provides real time control of the Motor System rather than just simply using the MCU because the MCU has many other data to process. The PIC is a power chip that has can perform m. The detailed operations how the PIC16F877 relate to this project are in the RTC Theories section below.

RTC Theories

Fig 10 PIC schematic



The PIC will be coded in C to implement the desired function. The related registers for PWM are

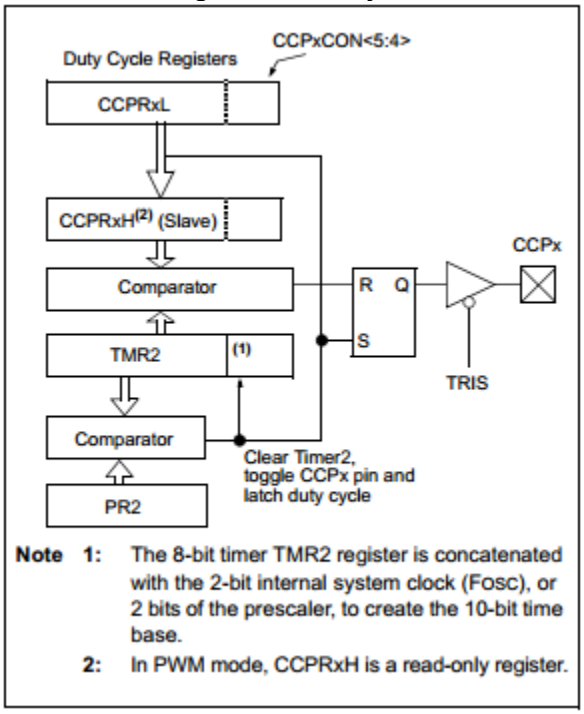
- PR2
- T2CON
- CCPRxL
- CCPxCON

The Fig. 11 below shows a simplified PWM block diagram from Microchip's datasheet. It demonstrates each cycle of how PWM operates. To output the correct signal from PWM, the period and the duty cycle (pulse width) are needed. They are calculated as follows in equation 10 and 11.

$$PWM\ Period = [(PR2) + 1] \bullet 4 \bullet TOSC \bullet (TMR2\ Prescale\ Value) \quad (10)$$

$$Pulse\ Width = (CCPRxL{:}CCPxCON<5:4>) \bullet TOSC \bullet (TMR2\ Prescale\ Value) \quad (11)$$

Fig 11 PWM Cycle



Video Module

Input: The Video Module's input is a webcam that is mounted on the front of the MV and captures the front of the MV. Its power will come from the USB port to the MCU.

Output: The output is the video signal captured by the webcam and sent to the MCU via an USB port. Additional video compressing, processing, and streaming are done in the MCU.

Description: An USB webcam QQ XTD is used to capture live video that will be used on our website to provide the users a real-time view of what's in front of the MV. The users can use the captured information to navigate the MV or simply look at the surroundings. Therefore, the webcam's video capturing quality has to be up to the standards. The QQ XTD will feature 5-12 FPS at 480x320 pixels depending on the bandwidth. In addition, the delay between the captured video to what the users see on the website has to be minimal.

Communication System

Input: The input is WIFI signal from routers in the MV's vicinity. It also receives power via USB port from the MCU

Output: The output is WIFI signal via USB port

Description: The Communication system is a Sabrent WIFI adapter that receives wireless signal from nearby routers. It is a necessary component because we need internet connection to connect to the user side. It follows the N standards about is backward compatible with 802.11b and g networks. This allows for versatile connections.

2.2.2 Software Blocks

The project's software is divided into two sections, Off-Board and On-Board. The Off-Board section refers to the user side; the On-Board section refers to the MCU on the Moving Vehicle. They are connected and send data through WIFI.

General Description

1. Off-board Software:

Web Browser:

This project supports any browser that follows modern html standards. This project will focus on implementation in Chrome.

WebServer:

Web hosting service is done through 2freehosting.com, which supports Apache HTTP Server, FTP, SQL, PHP, cURL, and Python.

AJAX IM/Html/ PHP scripts:

Scripts serve for three main purposes:

1. It is written to provide the web API for the client program. It sends the user command to the MCU and uploads MCU data to the server on user side.
2. It is written to provide the user basic user interfaces.
3. It is written to take the server's strong data processing ability to further analyzing the data.
4. It is written to provide WebRTC P2P data channel to setup P2P connection if the user has browser support.

2. On-Board Software:

There will be two programs responsible for the On-Board software, which are "CarClient" and "CarObAv". The Client program running in Linux environment on Raspberry Pi is called "CarClient". The Obstacle Avoidance System installed on PIC16F877 on Real Time Controller is called "CarObAv".

When programming on real-time critical system, we must prioritize reliability and performance. Reliability means the system shall be fault-tolerant. In this project, the car must deal with the non-stable WIFI signal and also deal with any errors generated during its movement. Performance means the system shall response to the real-time events as soon as possible. In this project, we will design a two-level control system: CarClient and CarObAv. CarObAv deals extensively with the real time processing. It takes inputs from the Raspberry Pi and from the Sensor Module, and then adjusts the speed and movement of the MV correspondingly.

The focus of this section is on CarClient because it interfaces the user side and the MV side. CarObAv depends heavily on Sensor System. Please refer to Sensor Module for the general algorithm to obstacle avoidance.

CarClient Description

Input:

User command data from the internet

Output:

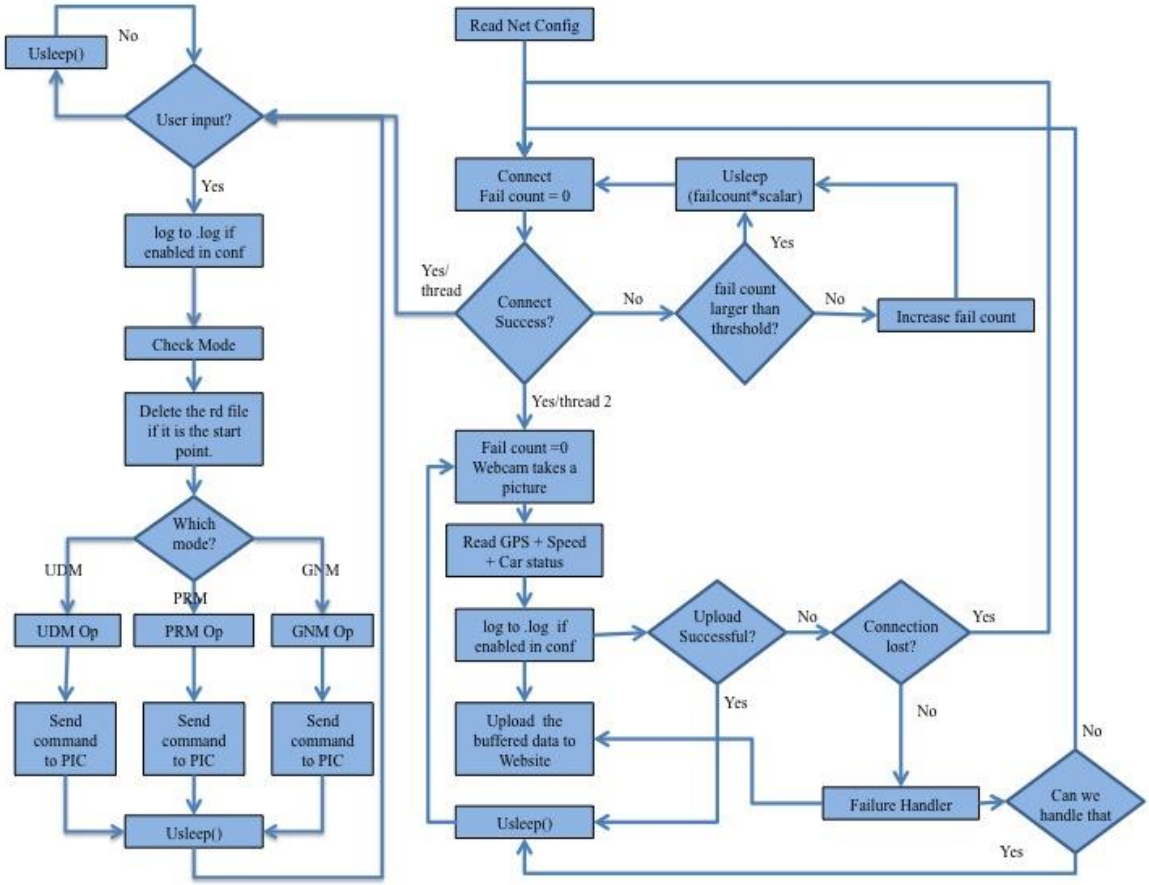
Video data, GPS location, and MV current ”Speed“

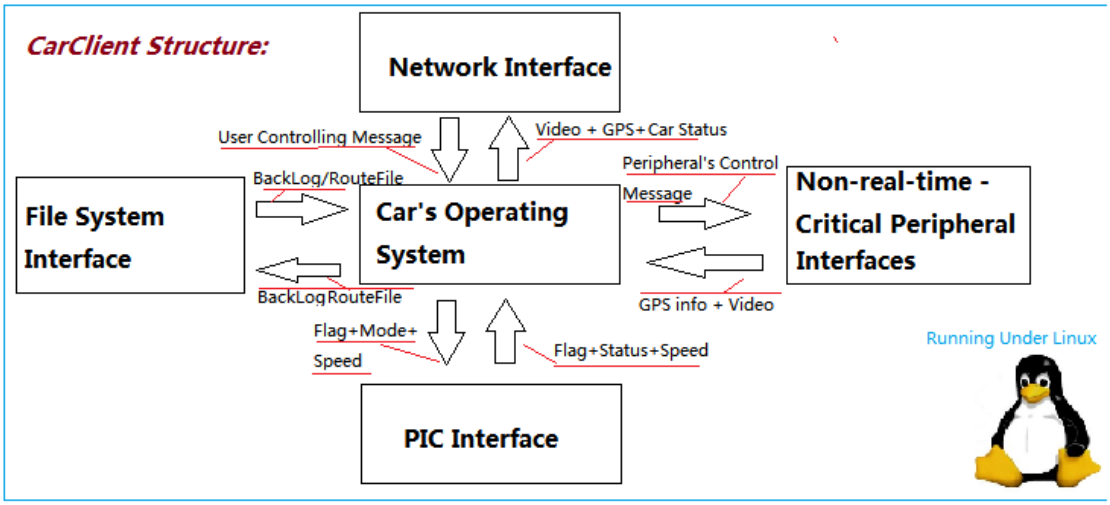
Environment:

CarClient is a daemon process running in LINUX after system boot. To initialize correctly, there must be another process running in background to roam the wireless signal. Setup is defined in /etc/interfaces, /etc/wpa-supplicant, and /etc/wpa_supplicant/wpa_supplicant.conf. It also requires an open-source webcam software fswebcam to be pre-installed in LINUX.

User must define a CarClientnet.conf configuration file where CarClient is installed to configure CarClient’s I/O channel and network routing schemes. In addition, in CarClientnet.conf, the user can specify two modes of operations regarding accessing internet. This will be explained Network Interface below.

Fig 12 Operation Flowchart and Structure





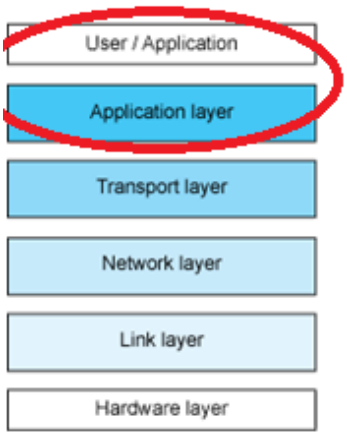
Note: Arrow indicates data flow. UDM: User Driving Mode; PRM: Program Routing Mode; GNM: GPS Navigation Mode;

Design Implementation and Philosophy:

Network Interface:

Network Interface is the core part of the Moving Vehicle (MV). The objective is to make the MV easily controlled in every device having access to internet. We first look at compatibility issues, which mean any controller should have the same agreement protocol to command the vehicle and receive the data from it. To simplify compatibility issues, web browser is chosen to minimize the work done to connect via internet.

Fig 13 Protocol Layers



In order to safely, reliably, and efficiently communicate via the internet, we need to realize many protocols. These protocols are built in layers in order to modularize the design. Every higher layer adds more features and takes use of the lower layer. The Fig 13 above illustrates the layers.

Linux kernel provides great convenience because it has already implemented most of the protocol suites and has a great ability scheduling tasks and handle network interrupt. The red circle in Fig 12 states our main focus of the implementation of network system. Table 4 below illustrates our project's network stack according to the layer in Fig 13.

Table 4 Network Stack

Application	CarClient / Browser
Application Layer	ICE/FTP/HTTP/
Transport Layer	TCP
Network Layer	IP
Link Layer	IEEE 802.11
Hardware Layer Layer	USB Physical layer

CarClientnet.conf:

CarClientnet.conf defines two mode of network interface operation. The user can specify a choice that is suitable for the user's browser support. One is the cURL mode, which lets

FTP and HTTP/POST to upload data and HTTP/GET to query user command. The other is Interactive Connectivity Establishment (ICE) mode, which sets the car as a P2P WebRTC client connecting to P2P data channel on the web browser. For the ICE mode implementation, we will use Chromium framework. Table 5 sums up the two modes.

Table 5 Difference between cURL and ICE

Mode	cURL Mode	ICE Mode		
		Libnice	Chromium (our choice)	Libjingle
Pros	• Compatible with any browser • Allow extra data processing in the cloud	• C library; Higher Performance • Full customer design	• Easy impmentation	• C++ library • easier than libnice implementation
Cons	• Extra Low performance	• Design Difficulties	• Low performance and efficiency on RPi	• Design difficulties

File System Interface:

There will be three kinds of data files that will be significantly handled in this project. They are:

1. Read/Write route files (.rt): Access under /route directory. It can be the backlog of the moving commands, pre-programmed route file or route file generated by the AI of the operating system
2. Read/Write image files (.jpg or .mpeg): Access under /video directory, camera captured image
3. Read/Write Log files (.log) : Access under /log directory. Recording of the system status.

Real Time Controller interface:

The RPi needs to communicate to the RTC's PIC. This will be done via direct wire connections between them. We will define two instructions. The data are transferred in the following fashion. (The pin numbers are with respect to the Raspberry Pi).

Output from RPi is an 8-bit instruction to PIC via pin 0-7. The format is:

2-bit-Mode||2-bit command||3-bit-speed||1-bit-flag

Input to the RPi is a 6-bit instruction coming from the PIC via pin 8-13. The format is:
2-bit-Status||3-bit-speed||1-bit-flag

Explanation of instruction:

- Flag bit-vector field indicate new instruction is coming
- Command bit-vector field indicates which direction is moving
- Speed bit-vector filed indicates at which speed the user expects the vehicle to move, but the car might not necessarily be moving because of obstacles. There are eight levels of speeds that the car can choose, 000 to 111, where 000 means no movement.

Table 6 summarizes the instructions

Table 6 Instruction Explanation

Mode	**Meaning**
00	Disable obstacle avoiding, full user control
01	Enable obstacle avoiding, half user control
10	-Optional-
11	-Optional-

Command	**Meaning**
00	Forward
01	Backward
10	Leftward
11	Rightward

Status	**Meaning**
00	Running normally
01	Discovers obstacles nearby
10	Cannot move
11	-Optional-

Car's Operating System (main()):

Its functions include:

1. A coordinator of the interfaces. Translate the messages exchanged between interfaces.
2. Generate .rd files. This includes:
    1) cURL to query the online database for routes
    2) Reverse the backlog route to make a recovery route file back to the start point.

Non-real-time-Critical Peripheral interfaces:

Non-real-time-Critical Peripheral interfaces are interfaces communicating with the camera (via USB) and the GPS module (Via Tx/Rx pins). The Car's OS would read the output generated by these components either directly from memory or from disk location where the components write to.

2.3 Simulation and Result

Motor System Simulation and Test

This is the simulation of our motor system and moving vehicle module. This simulation is programed in Quartus and the functionality of each part has been discussed in Motor System Theories on page 10.

Fig 14 is the code for our H-bridge simulation. Inputs A, B, C and D are the four gates of four MOSFETs. Outputs F_out and B_out are pins connected to positive and negative pins of the motors. Some inversions are made here for easier connection and programming and as a result, input pin A and D, B and C are connected together. The result would not vary with this inversion.

Simulation Code for motor 1 and motor 2 are shown below in Fig 15 and 16. Input p and n are abbreviation for positive and negative. CW and CCW are abbreviation for clockwise and counterclockwise.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4  entity mosfet is
5      port(   A: in std_logic;
6              B: in std_logic;
7              C: in std_logic;
8              D: in std_logic;
9              F_out: out std_logic;
10             B_out: out std_logic);
11 end mosfet;
12 architecture Behavioral of mosfet is
13 signal fwd, bwd: std_logic;
14 begin
15 process(A, B, C, D)
16 begin
17     if(A ='1' and D = '1' and B='0' and C='0') then
18         fwd <= '1';
19         bwd <= '0';
20
21     elsif(A ='0' and D = '0' and B='1' and C='1') then
22         fwd <= '0';
23         bwd <= '1';
24
25     elsif(A ='1' and D = '0' and B='1' and C='0') then
26         fwd <= '0';
27         bwd <= '0';
28
29     elsif(A ='0' and D = '1' and B='0' and C='1') then
30         fwd <= '0';
31         bwd <= '0';
32     else
33         fwd <= '0';
34         bwd <= '0';
35     end if;
36 end process;
37 F_out <= fwd;
38 B_out <= bwd;
39 end Behavioral;
```

Fig 14 H-bridge simulation code

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.NUMERIC_STD.ALL;
4
5   entity m1 is
6       port(   p: in std_logic;
7               n: in std_logic;
8               cw: out std_logic;
9               ccw: out std_logic);
10  end m1;
11
12  architecture Behavioral of m1 is
13  signal cww, ccww: std_logic;
14  begin
15  process(p, n)
16  begin
17      if(p='1' and n='0')then
18          cww <= '0';
19          ccww <= '1';
20
21      elsif(p='0' and n='1')then
22          cww <= '1';
23          ccww <= '0';
24
25      else
26          cww <= '0';
27          ccww <= '0';
28      end if;
29  end process;
30
31  cw <= cww;
32  ccw <= ccww;
33  end behavioral;
```

Fig 15 Motor 1 simulation code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity m2 is
    port(   p: in std_logic;
            n: in std_logic;
            cw: out std_logic;
            ccw: out std_logic);
end m2;

architecture Behavioral of m2 is
signal cww, ccww: std_logic;
begin
process(p, n)
begin
    if(p='1' and n='0')then
        cww <= '1';
        ccww <= '0';

    elsif(p='0' and n='1')then
        cww <= '0';
        ccww <= '1';

    else
        cww <= '0';
        ccww <= '0';
    end if;
end process;
cw <= cww;
ccw <= ccww;

end behavioral;
```

Fig 16 Motor 2 simulation code

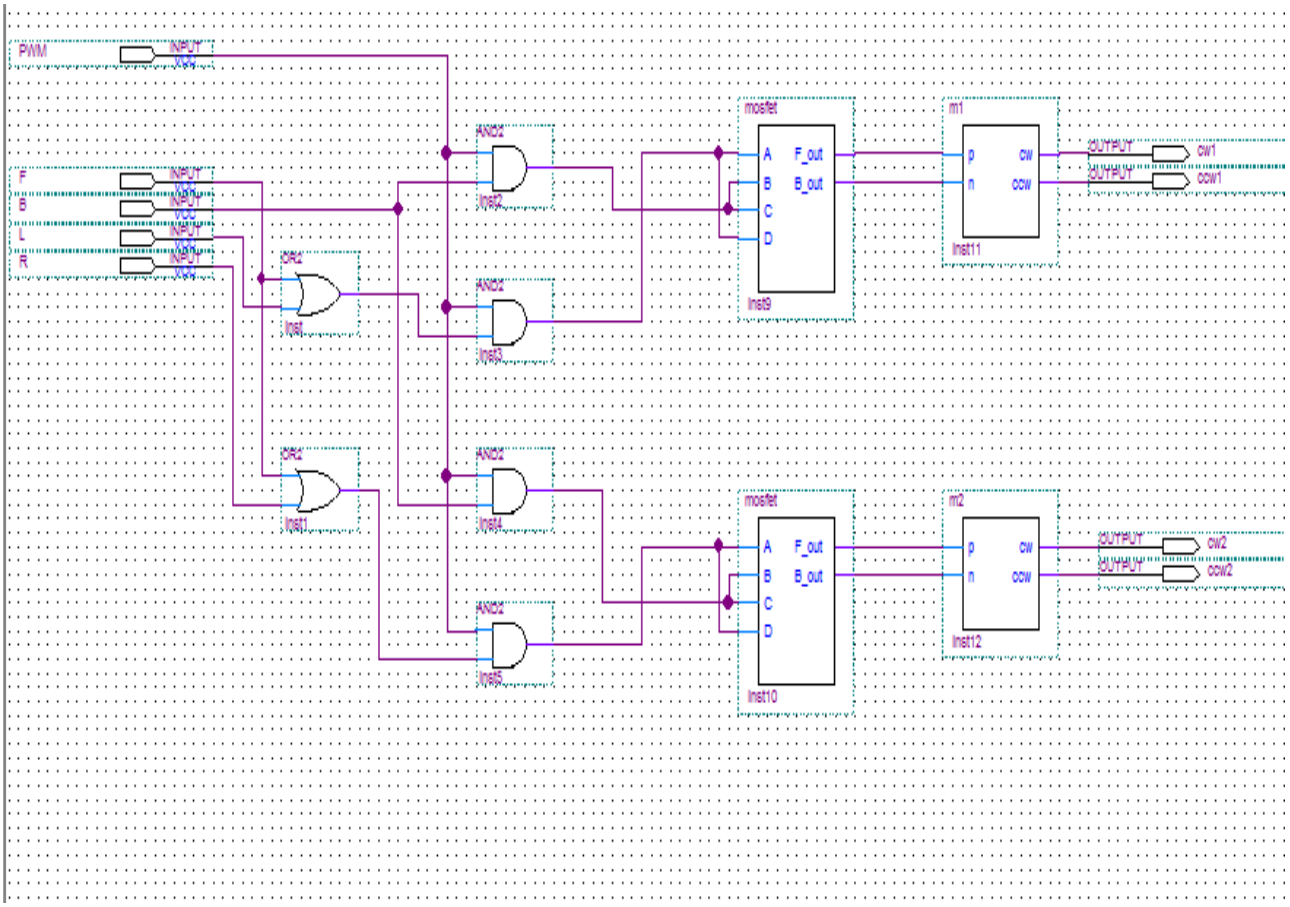


Fig 17 Motor schematic diagram

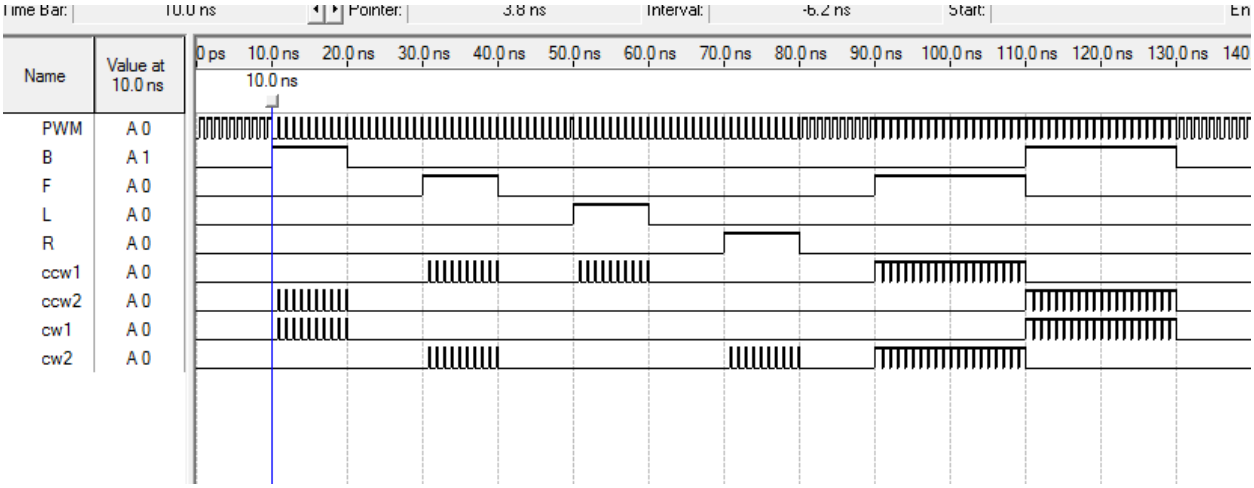


Fig 18 Motor simulation waveform

From the simulation waveform we can clear see: when backward is a high, M1 turns CW and M2 turns CCW; when forward is a high, M1 turns CCW and M2 turns CW and etc. Additionally, if PWM clock cycle changes, the M1 and M2 spinning cycle will also change. All waveform results are as expected so our design in motor system and moving vehicle are correct.

After the simulation, we also tested the Motor System to see the actual power delivered to the Motor System.
The datasheet specifies that the motors are rated at 4.5-6.0V with a no load current of 250mA and stall current of 1.8A at 4.5 each.
When the two DC motors are connected without the H-Bridge, the power rating is measured and calculated by the following steps. The AA batteries are directly connected to both motors, which are connected to the gears. The voltage is at 3.78V +/- 0.01V when the motor is running; 4.38V when standing by. The current when both motors are running is at 610mA +/- 30mA. Therefore, the power delivered directly to the motor when the motor is running is calculated as following:

$$Total\ power\ of\ both\ motors = V \times I = 3.78V \times 610mA = 3.29\ W \pm 0.12 \quad (12)$$

Sensor Simulation

Sensor simulation is performed by measuring the output voltage level of LV-MaxSonar-EZ pin AN. The result is shown in figure 19 and figure 20. From measurement, the lowest voltage level for EZ family is about 6.3mV, which is about 16cm by using equation (7). Thus the smallest range of the ultrasonic detector is about 16cm. We also measured the voltage level after we put an obstacle around 2.2 meters away from the sensor. The average voltage level changed to 881.5mV, which is about 2.28 meters. The range detection error is about (2.28-2.2)/2.2= 3.6%.The voltage level and range have a linear proportional relationship and it can be observed from the oscilloscope.

Fig. 19 Sensor at 2.2 meters

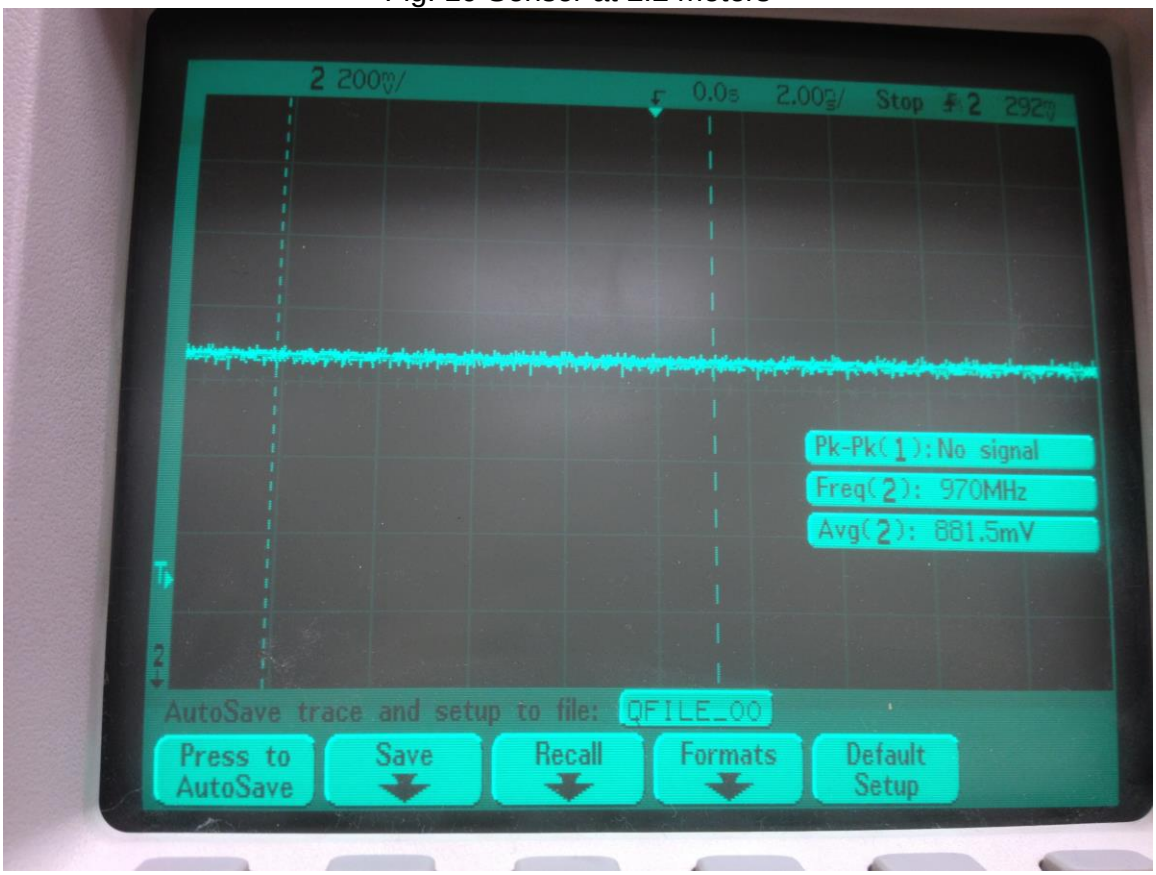
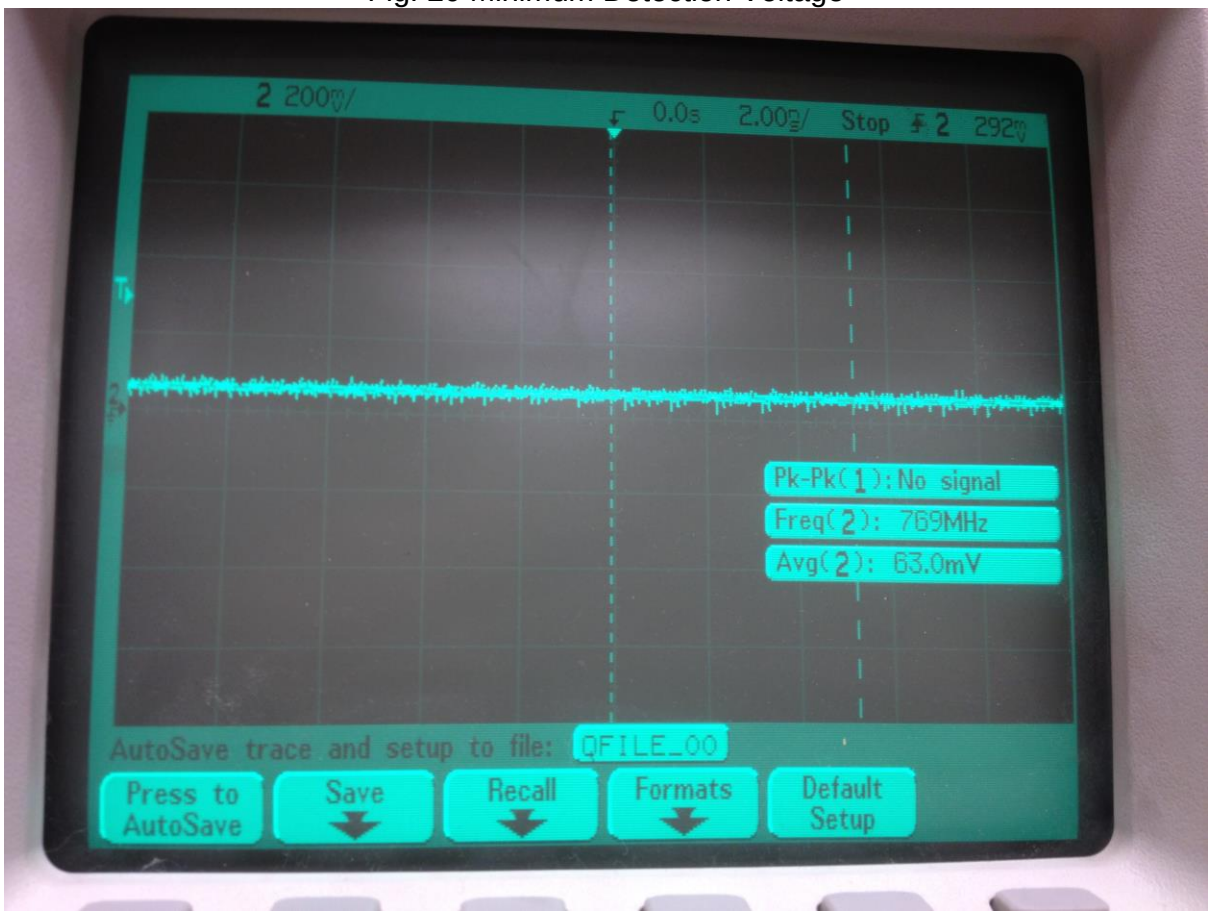


Fig. 20 Minimum Detection Voltage



GPS navigation simulation and Results

To simulate the surface distance calculation algorithm, we used equation (8) and (9) in MatLab to find out the distance between two points with coordinate: 40.7142 N, 74.0064 W and 40.7142 N, 74.0066 W. The result of our simulation showed the distance between those two points is 16.9 meters. Comparing this result with Google Map, which gave us 49 ft =14.9 meters, the error would be (16.9-14.9)/14.9=13%. This is a big error but for small distance navigation in this project, this result is within tolerance. The coding for this simulation is shown in figure 21 and the results are shown in figure 22 and 23

Fig 21 Simulation Code for Surface Distance Algorithm

file can be published to a formatted document. For more information, see the publishing video or help.

```
%% GPS simulation
function [dis]= gps(lat1, long1, lat2, long2)
r=6371.009;
a=((sind((lat2-lat1)/2))^2+cosd(lat1)*cosd(lat2)*(sind((long2-long1)/2))^2);
dis=2*r*asin(sqrt(a));
```

Fig 22 Simulation Result for Surface Distance Algorithm

```
>> gps(40.7142, -74.0064, 40.7142, -74.0066)

ans =

    0.0169
```

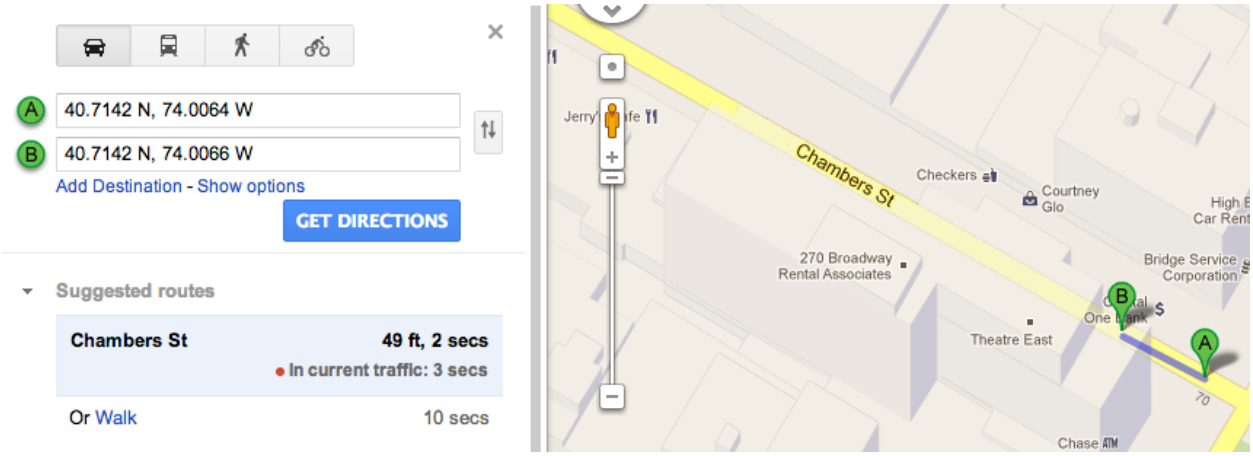


Fig 23 Google Map Result for Surface Distance

3.0 Requirements and Verification

3.1 Project requirement and verification table

Requirement	Verification
1. Power Module **<u>1.a</u>** Three AA Batteries provide enough energy to run the motor at maximum load for at least 10 minutes **<u>1.a.1</u>** Motor's peak power should be less than 2*0.9*3.75 =3.375W **<u>1.a.2</u>** The output voltage needs	**1. Power module** **<u>1.a</u>** From Power Budget section, the three AA batteries have 9.9 Wh. The requirement passes If total peak power of subcomponents adds up to be less than 9.9W. **<u>1.a.1</u>** Measure the motor's current and voltage using multi-meter (Fluke 12) with maximum loads running on pile floor, and then multiply the current and voltage to get peak power, which should less than 3.375W **<u>1.a.2</u>** Use NI DAQ 9207 to take the voltage output for 10

to be above 3.75V for the duration of 10 minutes	minutes running at maximum loads and on pile floor, and then examines the acquired data on LabView.
1.a.3 Output voltage's ripple from the AA batteries should vary within +/- 10mV	**1.a.3** Use NI DAQ 9207 to take of the voltage output for 10 minutes running at maximum loads and on pile floor, and then examines the acquired data on LabView.
1.b Microcontroller power pack provides enough energy to run the RPi to run at maximum power for at least 10 minutes while providing at least 700mA at 5V.	**1.b** From the Power Budget section, the MHK power pack has 22 Wh. The requirement passes if total peak power of subcomponents adds up to be less than 22W.
1.b.1 RPi's peak power should be less than 3.5W	**1.b.1** Use Tektronix TM504 instrument to measure the USB power input pin of RPi while performing data streaming on RPi.
1.b.2 Each sensor's total peak power should be less than 15 mW	**1.b.2** Use multi-meter to measure the voltage and current at the "Vcc" pin while letting the sensor detect a close ranged object.
1.b.3 PIC's peak power should less than 10 mW	**1.b.3** Use Tektronix TM504 instrument to measure the power input pin on PIC16F877 while reading signals from Sensor Module
1.b.4 MHK battery pack provides at least 700 mA at 5V +/- 10mV for the duration of 10 minutes	**1.b.4** Use DAQ 9207 to take voltage output for 10 minutes running RPi, sensor, and PIC at maximum loads.
2. Motor system Module	**2. Motor system module**
2.a Motor is not damaged under supply voltage and current	**2.a** Motors performance correctly is the basic requirement; ensure using three new AA batteries to test. Disconnect motors from circuit, then draw two wires from positive and negative sides of the battery carrier. Connect motors directly to AA batteries. Check and make sure motor could rotate; otherwise motors are damaged.

2.b Ability to operate continuously between 50%—100% duty cycle; requirement will be verified if all sub-requirements are verified:	**2.b** Ensure requirement **2.a** is verified; turn on vehicle and turn on microcontroller.
2.b.1 All gear teeth are fully engaged.	**2.b.1** No disconnection between gears will guarantee motors not slip at 100% duty cycle. Slowly rotate each tire by hands to check if there is any disconnection at gears by listening and observation
2.b.2 Gate voltage at each MOSFET must be larger than 3.3V	**2.b.2** Gate voltage higher than 3.3V will guarantee current flow across H-bridge; use multi-meter (Fluke 12) to test. Draw wires from pin 3 and 9 of each H-bridge. Draw a wire from RPI ground. Connect the positive port of the multi-meter to this wire and connect the negative port to common ground. Measure voltage and record data. Measurement should always be larger than 3.3V
2.b.3 PWM pin is able to output a square wave with 100% duty cycle	**2.b.3** A 100% duty cycle square wave is necessary to provide full PWM; use oscilloscope (Agilent 54642A) to test. Setup PWM at 100% duty cycle by set digital duty cycle at binary 1111. Then draw a wire from PWM output from PIC unit. Draw a wire from RPI ground. Connect positive wire to PWM output and negative wire to common ground. Press ‘auto-scale’ on oscilloscope. Press ‘quick measurement’ and chose “duty cycle”. Measurement should be 100% +/- 0.1%.
2.b.4 PWM should have minimum frequency of 500Hz	**2.b.4** Minimum frequency of 500Hz will guarantee motor run continuously (without clicking); use oscilloscope (Agilent 54642A) to test. Draw a wire from PWM output from PIC unit. Draw a wire from RPI ground. Connect positive wire to PWM output and negative wire to common ground. Press ‘auto-scale’ on oscilloscope. Press ‘quick measurement’

	and chose ‘frequency’. Measurement should have minimum of 500 Hz.
2.c Operate within reasonable temperature at maximum duty cycle (10 degrees +/- 10% above ambient temperature	**2.c** Ensure requirement **2.a** and **2.b** are verified; temperature limit must be verified to protect motor from damaging by heat. Set up PWM at maximum (100%) by set the digital duty cycle at binary 1111. Attach a mercury thermometer at the surface of the motor with tape. Measure the motor surface temperature before motor is turned on. Turn on motor and let it run until battery dies out. Record the temperature of the mercury thermometer, which is the maximum operation temperature. Measurement should not be 10 ± 10% degrees above ambient temperature
2.d React to control signal within reasonable lagging tolerance (less than 0.5 +/- 0.1s)	**2.d** Ensure requirement 2.a and 2.b are verified; lagging tolerance must be verified to guarantee user-friendly control experience. Log in to our website and click User Driving Mode. Enter “w” and Server starts a stopwatch at t1. Server stop stopwatch at t2 when it receives the request from the car. Round Trip Time (RTT)=t2-t1. The RTT should not exceed 0.5 ± 0.1s
3. Navigation System	**3. Navigation System**
3.a The Navigation System is functional on MV	**3.a** Read out input signal from PIC16F877, which returns if a signal has been received
3.b User should be able to use auto-navigation mode.	**3.b** Ensure Internet accessibility on the MV; checks users’ browser for Google Map support. User enters a location, the MV should move toward that direction
3.b.1 GPS should be able to accurately track the moving platform within 10 meters accuracy	**3.b.1** Using GPS Navigation Mode, record longitude and latitude and enter a new long. And lat. within 2’’. After the MV comes to a stop, record its new location. Finally, check the accuracy via Google Map. The difference should be within 10 meters.

4. Webcam **4.a** Webcam with USB connection is able to perform correctly **4.b** Webcam should provide at least 10 frame/s at resolution less than 640X480 to guarantee fluent vision	**4. Webcam** **4.a** Webcam working normally is the basic requirement; ensure using a computer with Windows XP or higher version to test. Connect webcam to the computer USB port. Windows will automatically detect webcam as new hardware. Click “Open new hardware” at the prop-up interface. Webcam should be started; check the camera window on the computer screen to see whether it is working **4.b** Ensure **4.a** is verified and connect webcam back to our project. Type ‘Sudo apt-get fswebcam’ to Run fswebcam(version 20110717) in shell : Fswebcam –r 640x480 –d /dev/video0 test.jpg –fps 10 Fswebcam –list-framesizes We can see the camera can support up to 640x480 and 10 frames/s is barely on the edge of the camera’s ability. Grab image larger than 480*320 I.e. 960*480 can clearly see the webcam fails to respond.
5. Sensor system **5.a** Sensor System should function and respond to obstacles. **5.a.1** Sensor should detect obstacles’ range accurately (+/- 2%) down to minimum of 10 cm. Below 10 cm, the DC level should stay within +/- 0.1%. **5.a.2** Sensor should detect obstacles’ range accurately (+/- 2%) up to maximum of 50 cm. The distance away from sensor	**5.Sensor system** **5.a** Supply 5 V to Vcc of a sensor; connect AN output to oscilloscope, and then move a book back and forth in front of the sensor. The DC level on the oscilloscope should decrease and increase. **5.a.1** Connect AN output of sensor to oscilloscope; then carefully bring a book from about 15 cm toward the sensor. Observe the DC level on the oscilloscope as the distance of the book approaches 10 cm. When It’s at about 10 cm, the DC level should reflect the range within 2%. The reading should stay within +/- 1% of its DC level when the distance is less than 10 cm. **5.a.2** Connect AN output of sensor to oscilloscope; then carefully bring a book from about 15 cm away from the sensor. Observe and record the DC level on the oscilloscope. When the book is at about 25 cm away,

should reflect linearly on the oscilloscope reading at Vcc/512 per inch **5.b.1** The Sensor System should correspond with the RTC at maximum of 1 msec (immediate)	check the DC level. It should be about 39 mV +/- 2% higher the original. In addition, when the book is held steady, the ripples of DC level should stay within +/- 0.1% **5.b.1** Connect Sensor System to the RTC and power on both. Move a book back and forth from the sensor. Set a timer program in PIC to record the reaction time and output to RPi. Should be almost immediate to human reaction time (<1 msec).
6. Microcontroller **6.a** RPi powers on and functions as normal **6.b** WIFI adapter functions and passes information between RPi's server and user's server **6.c** SD memory card transfer rate is higher than class 4 and memory is larger than 4GB	**6. Microcontroller** **6.a** Plug in power supply for RPi, and make sure it's powered on. **6.b** User sends a Turn Left command on the website, and checks the MV's movement to make sure it takes a left turn. **6.c** Insert SD card into PC and checks the storing capacity. Copy a large file to check the average transfer speed to be higher than 4 MB/sec.
7. Website **7.a** Website should be functional. **7.a.1** The user can create account and log in to take control a of an available MV	7. **Website** **7.a** User can go to the website afallon.yzi.me **7.a.1** User goes to the website, creates an account, and then finds an available MV to control.

3.2 Tolerance Analysis

The delay time between the website to the MV is the most important factor of our design. There will exist delay time between the MV and the website as well as delay time between the microcontroller and its attached components. Having immediate response from the MV is desired for the users' experience. For example, if a user enters a command on the website to navigate the MV forward, the command should reach the MV and performed by the MV within one second or less. In addition, having a short delay time between the microcontroller and its component will ensure the safety of the MV. For example, if the

sensors see an obstacle, the MV should respond quickly to avoid potential hazards.

Requirement	Verification	Tolerance Analysis
To have delay time between our web server to MV less than 0.5 sec	Send various signals from the website and time the response at the MV. In addition, we will test the response time of the signals for each module.	If faster, then we don't need to worry. Thus, it can be only slower than 0.5 + 0.2 sec at max

4.0 Cost and Schedule

4.1 Cost

4.1.2 Labor cost

Name	(Rate/hour)×(Total Hours per week)×12	Total Price
Yigao Shao	$\$35 \times 30 hours \times 12\ weeks$	$12,600
Kecheng Liu	$\$35 \times 30 hours \times 12\ weeks$	$12,600
Yubo Liu	$\$35 \times 30 hours \times 12\ weeks$	$12,600
Labor total		$37,800

4.1.2 Part Cost

Part Name	Manufacture	Model /Part#	Description	Quan	Price /Unit	Total Price
Mr. Basic	DAGU	TR3	Vehicle	1	$30	$30
Raspberry Pi	Farnell and RS Component	Model B	Microcontroller	1	$35	$35
GPS	Global Sat	EM-408	GPS	1	$30 (445)	$30
Webcam	Tencent	QQ XTD	Webcam	1	$25	$25
WIFI Adapter	Sabrent	A11N	WIFI	1	$20.25	$20.25
Ultrasonic Sensor	MaxBotix	MB1000 Family	Sensors	5	$28 (445)	$140
AA battery	Energizer	Alkaline	Rechargeable	6	$3	$18
MHK Batter	MHK	Power Pack 2200mAH	Battery	1	$20	$20

PIC	MicroChip	PIC16F877	PIC	1	$4.50 (445)	$4.5
SD Card	SanDisk	008G-A11	Memory	1	$18	$18
IC & transistors					(445)	$10
Resistor, caps					(445)	$15
PCBs			Fabricated by Part Shop	3	$0	$0
					Total	$355.75

4.1.3 Total Cost

Part Cost	$365.75
Labor Cost	$37,800
Total Cost	$38,165.5

4.2 Schedule

Week	Yigao	Yubo	Kecheng
1/14	Discussion about project ideas, talk about ordering parts	Discussion about project ideas, talk about ordering parts	Discussion about project ideas, talk about ordering parts
1/21	Discussion about project ideas Set up server and domain	Discussion about project ideas Research about radar and ultrasonic system	Discussion about project ideas Research about H-Bridge and PWM
1/28	RFA and proposal on software	RFA and proposal, focus on sensors and organizing	RFA and proposal, focus on power and GPS
2/4	Set up video streaming on website, RPi I/O configuration, order parts	Get PWM and H-Bridge onto PCB	H-Bridge schematic and design, order part for H-Bridge
2/11	Interfacing between the software and hardware	Ultrasonic sensor design, and get parts	Finish the power system, and test for robustness
2/18	Implement the speed control's software	Test the ultrasonic sensors and install on PCB (headers)	Research about GPS
2/25	Design Review, interface between the RPi and PIC	Design Review, Code PIC for PWM	Implement GPS and study algorithms for automatic navigation
3/4	Finish up internet reliability problem	Test and verify sensor system with Motor System	Design Review, study algorithm and code on RPi
3/11	Individual Report	Individual Report	Individual Report
3/18	Spring Break	Spring Break	Spring Break
3/25	Test internet reliability and speed optimization	Test for robustness of the obstacle detect and avoidance system	Test and verify GPS system
4/1	Debug/Test the website, work on path memorization	Debug/Test the user navigation scheme	Debug/Test the Automatic navigation scheme
4/8	Test the whole software system, Final Paper, focuse on software	Final Paper, focus on sensors and detection, and organizing the whole paper;	Final Paper, focus on navigation, control, and power
4/15	Final Paper, focus on software side prepare for demo and presentation	Final Paper, prepare for demo and presentation. Finishing up and organize the whole paper	Final Paper, prepare for demo and presentation. Put in the necessary figures, schematics, and results
4/22	Demo	Demo	Demo
4/29	Presentation	Presentation	Presentation

5.0 Ethics

The purpose of this project is to develop a real-time Wi-Fi controlled moving platform. Its Internet remote control ability ensures that the vehicle could be used in places where potential hazards for human beings might occur. With such function, our device helps increase the safety and health of the user, which is consistent with the first code of the IEEE Code of Ethics:

1. To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment.
2. Throughout the development of the platform, we will only make claims and estimates based on real data acquired from measurements. We will be honest and will not falsify the data acquired from our test procedures, as cited in the third code.
3. To be honest and realistic in stating claims or estimates based on available data.
4. After this project, we will have learned various real-world industrial systems such as the PIC emulation, Internet security and certification agreement, and the Raspberry Pi protocol usage. From experiments and testing procedures, our understanding of technology and comprehension of real-world application will be improved, as directed in the 5th and 6th codes of the IEEE Code of Ethics.
5. To improve the understanding of technology; its appropriate application, and potential consequences. To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.
6. While working on the project, we accept criticism and suggestions from both group members and from outside. Additionally, since this project is a group project, we will help each other in his academic development and to support him in following the code of ethics.
7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.
8. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

6.0 Safety

Safety issue is very important because the work we performed in a laboratory can have a significant effect on us now and in the future. Our knowledge of safety is a critical element in protecting ourselves from potential hazards. Others who work in the laboratory—co-workers and service personnel—also rely on us to make choices that keep them safe.
All team members in our group have finished the *General Laboratory Safety Training* and *Electrical Safety for Labs* and have acquired *Certificate of Completion* provided by *Division of Research Safety*. We will closely follow the laboratory rules through design and be responsible for our own safety.
We will always keep safety in mind because the key to protecting our health and safety is to be able to identify hazards and take necessary precautions to protect ourselves and other people every time we work in lab.

In addition, the MV will be moving around the Engineering Quad, which could be a potential hazard to others and to itself. Thus, we will have a flag on the MV as an indicator to others. We will be using lab equipment such as oscilloscope. We should be careful as to not eat or drink around the testing equipment. We should not make circuit changes or perform wiring when connected and power is on. Changes should be performed while power is off. In addition, we will be attentive with readings of the max input to the oscilloscope.

7.0 References

Beej's Guide to Network Programming. [Online]. Available: http://beej.us/guide/bgnet/

Chormium Community. [Online]. Available: http://www.chromium.org/

Division of Research Safety (DRS) [Online]. Available: https://www.drs.illinois.edu/tdb2/Quizzer/Presentations/GLS/GeneralLabSafety.html

Eagle User Guide,Cadsoft, 2010, [Online]. Available: http://www.cadsoft.de/wp-content/uploads/2011/05/manual_en.pdf

EM408 GPS Engine Board, Global Sat, Taiwan, 2009. [Online]. Available: http://www.usglobalsat.com/store/download/47/em408_ug.pdf

GCC Community. [Online]. Available:http://gcc.gnu.org/

HI-TECH C Compiler, Microchip Technology Inc, AZ, 2010. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/manual_PICC_983.pdf

ICE protocol RFC 5245, IETF, J.Rosenberg et.al, 2010,[Online].Available:http://tools.ietf.org/html/rfc5245

IEEE Code of Ethics [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html

Introduction to PIC Programming: [Online]. Available: http://www.gooligum.com.au/tutorials/midrange/PIC_Mid_C_1.pdf

Libcurl Community. [Online]. Available: http://curl.haxx.se/libcurl/c/libcurl-tutorial.html

Libjingle Community. [Online]. Available: https://developers.google.com/talk/libjingle/index

Libnice Community.[Online]. Available: http://nice.freedesktop.org/wiki/

LV-MaxSonar-EZ0 High Performance Sonar Range Finder, Maxbotix, MN, 2005. [Online]. Available: http://www.maxbotix.com/documents/MB1010_Datasheet.pdf

LV-MaxSonar-EZ1 High Performance Sonar Range Finder, Maxbotix, MN, 2005-2012. [Online]. Available: http://www.maxbotix.com/documents/MB1000_Datasheet.pdf

MPLAB tutorial, Inbredded Systemshttp. [Online]. Available://morrish.ca/beginner/interrupts.php

MPLAB X Integrated Development Environment, Microchip Technology Inc, AZ,2011-2012.[Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/52027B.pdf

Mr.Basic Model TR-3, DAGU Hi-Tech Electronic Co. LTD, China, 2009. [Online]. Available: http://www.robotshop.com/content/PDF/mr-basic-tr-3.pdf

PIC16F87X7 40-Pin 8-Bit CMOS FLASH Microcontroller, Microchip Technology Inc, AZ, 2001.[Online]. Available http://ww1.microchip.com/downloads/en/devicedoc/30292c.pdf

Raspberry Pi community. [Online]. Available: http://elinux.org/RPi_Community

Raspberry Pi Model B revision, University of Cambridge¡¯s Computer Laboratory, UK, 2012. [Online] Available:http://www.raspberrypi.org/wp-content/uploads/2012/12/quick-start-guide-v1.1.pdf and http://elinux.org/RPi_Community

STUN protocol (bis) RFC 5389, IETF J. Rosenberg et.al, [Online]. Available: http://tools.ietf.org/html/rfc5389

STUN protocol RFC 3489, IETF, J. Rosenberg et.al, 2003, [Online]. Available: http://tools.ietf.org/html/rfc3489

The LLVM Compiler Infrastructure Project. [Online]. Available:http://llvm.org/

TURN protocol RFC 5766,IFTF, R. Mahy et.al. 2010. [Online]. Available: http://tools.ietf.org/html/rfc5766

WebRTC community. [Online]. Available: http://www.webrtc.org

WebRTC tutorial, HTML5 Rocks, [Online]. Available:http://www.html5rocks.com/en/tutorials/webrtc/basics/

XC8 Compiler, Microchip Technology Inc, AZ. 2012. Online. Available:http://ww1.microchip.com/downloads/en/DeviceDoc/52053B.pdf