University of Illinois at Urbana-Champaign

ECE 445 Spring 2013

Senior Design

Design Review

Aggressive Chasing Car

Group 38

Members: Hai Chi, Zhe Ji

TA: Mustafa Mukadam

Table of Contents

1. Introduction & General Description

1.1 Original Motivation

Senior design is a class where we make our dreams come true. The dream might not be an ambitious one such as becoming a billionaire, but this is a critical start of everything before our successful careers. Through the project we develop and improve our social and technical skills, which are both very important to our future. For this project, when we were bouncing the ideas of the project, we found out both of us have a keen on chasing running target. It seems that both of us want to be the "good guys" and love classic movie scenes such as chasing the bad guys. Luckily we have our idea approved, and now it's the chance for us to direct a mini chasing scene ourselves!

1.2 Objective and Scene Description

Our goal is to design a linkage system among a running car (the bad guys) remotely controlled by us, a chasing car (the policeman car) driven by a microcontroller with an image processing component and pursuit-evasion algorithm, and the camera on the ceiling (the satellite, or the chopper).

At first, the chasing car hides itself in a hidden dark corner and monitors the cars passing in front of it using sensors in front of it. Once it detects the target, the running car, it starts to chase behind it with a relatively higher speed. It's of course very possible that the chasing car loses the running car in site, possibly because the running car has gone out of range, or blocked by some obstacles. In either situation, the chasing car will call for help from a satellite or a helicopter. From the view above, the camera will provide a general direction for the chasing car to overtake the running car. But for the chasing car, it is its own job to decide which direction to

go, along with detecting the blocks and obstacles on its way and bypassing them. Once the chasing car spots the running car again, the chasing car should switch back to sensor-based view. The game ends when the chasing car surpasses the running car and stops in front of it.

1.3 Features and Functionalities

Running car:

- Wirelessly, manually controlled.
- Smart and cunning.
- It’s a Mustang!

Chasing car:

- Trajectory calculation and estimation.
- Wireless signal receiving and transmitting
- Microcontroller-driven movement.
- Obstacles dodging.
- Surpassing and intercepting.

Camera:

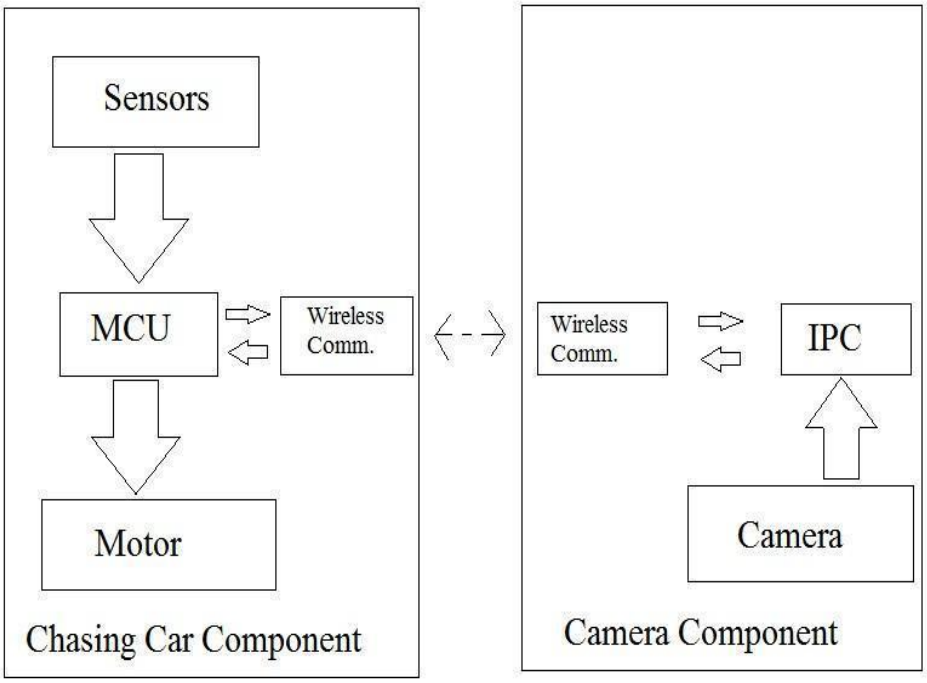
- Image recognition and processing.
- Trajectory calculation and interception coordinate estimation.
- Wireless signal receiving and transmitting

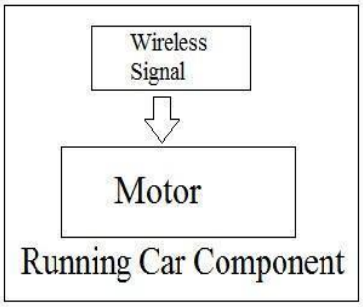
1.4 Benefits

This is a project of our own interest. It does not provide any kind of benefit to our society. But it is really a good experience and a strong bullet on our resume. It examines and integrates what we learned in classes and put it into practice. It will help improve our abilities to design, abilities to learn new things, debug, writing papers and documentation. It also teaches us how to collaborate, communicate, help each other and share ideas. All of them mentioned above are the very keys to our success in the future.

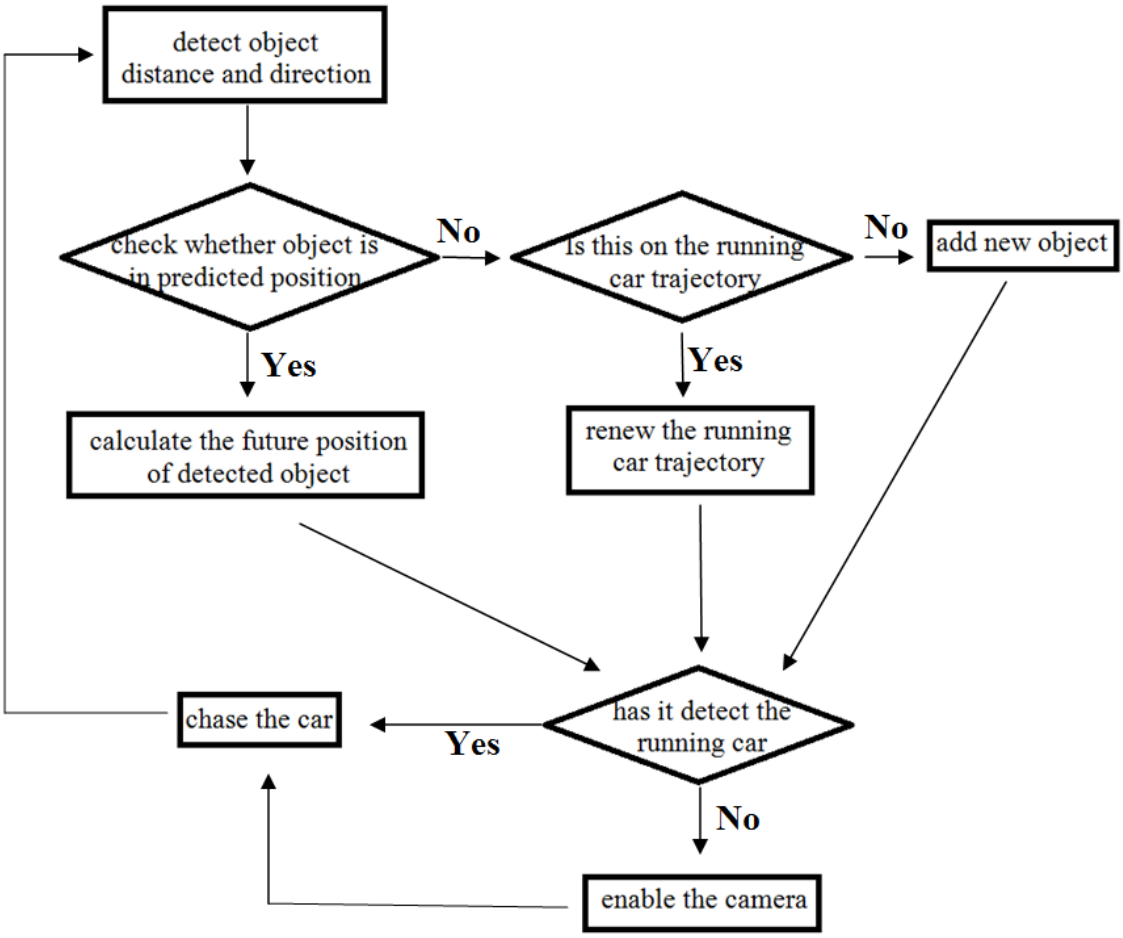
2. Design / Schematics

2.1 Design Block Diagram





2.2 MCU Logic Flow Chart



2.3 Camera Component

2.3.1 Camera

Input: The chasing scene.

Output: 30 fps 640 * 480 jpeg pictures.

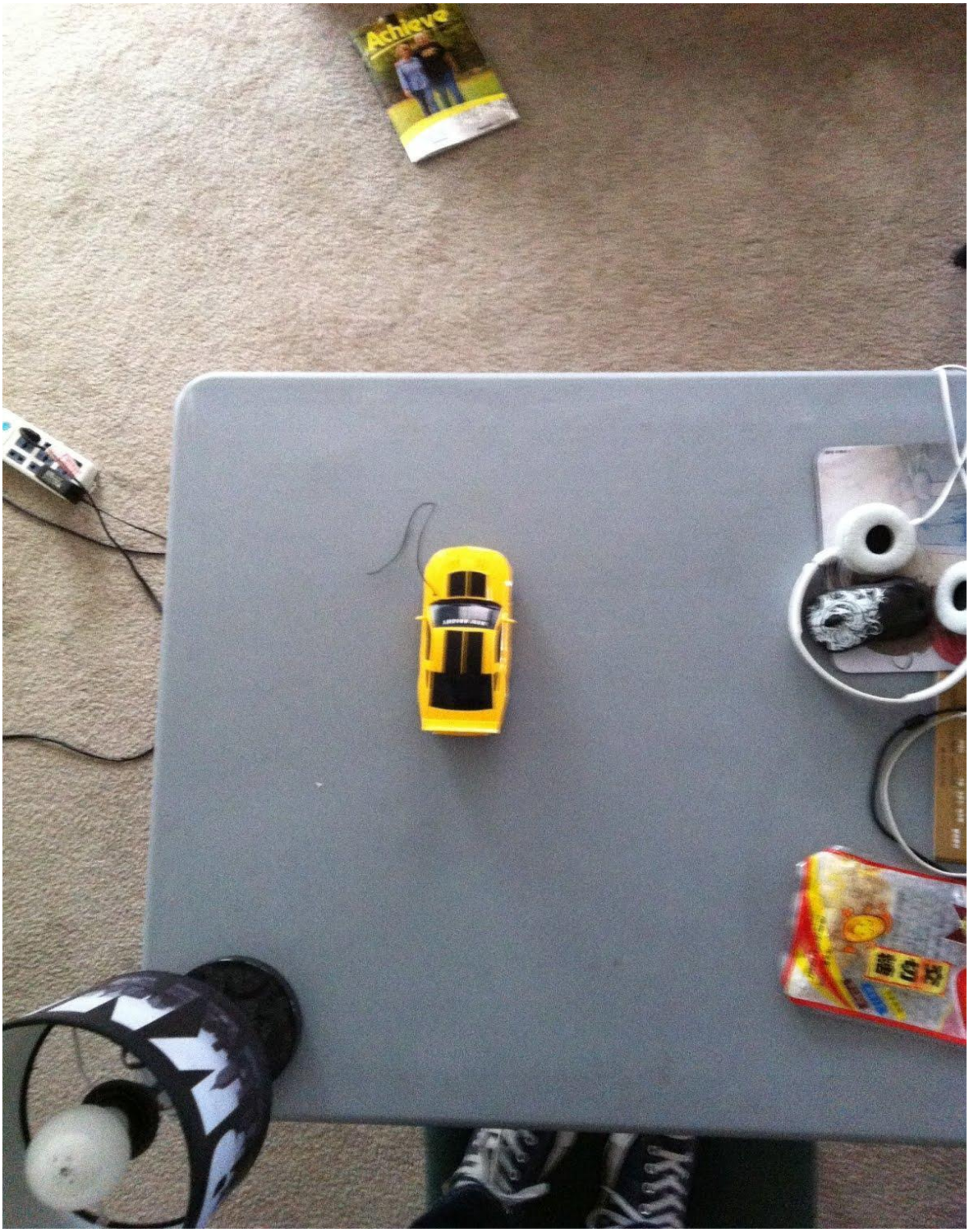
Description: The camera purchased is a Fire-i digital camera. It has a maximum frequency of 30Hz and a resolution of 640x480. The camera will communicate with the CVS (compact vision system) via a FireWire connection. It will be positioned on the ceiling. The orientation of the camera will be vertically downward such that it is able to focus on the motions on the ground.

2.3.2 Image Processing Component (IPC)

Input: Multiple dimension 640 * 480 jpeg pictures with the chasing scene in it.

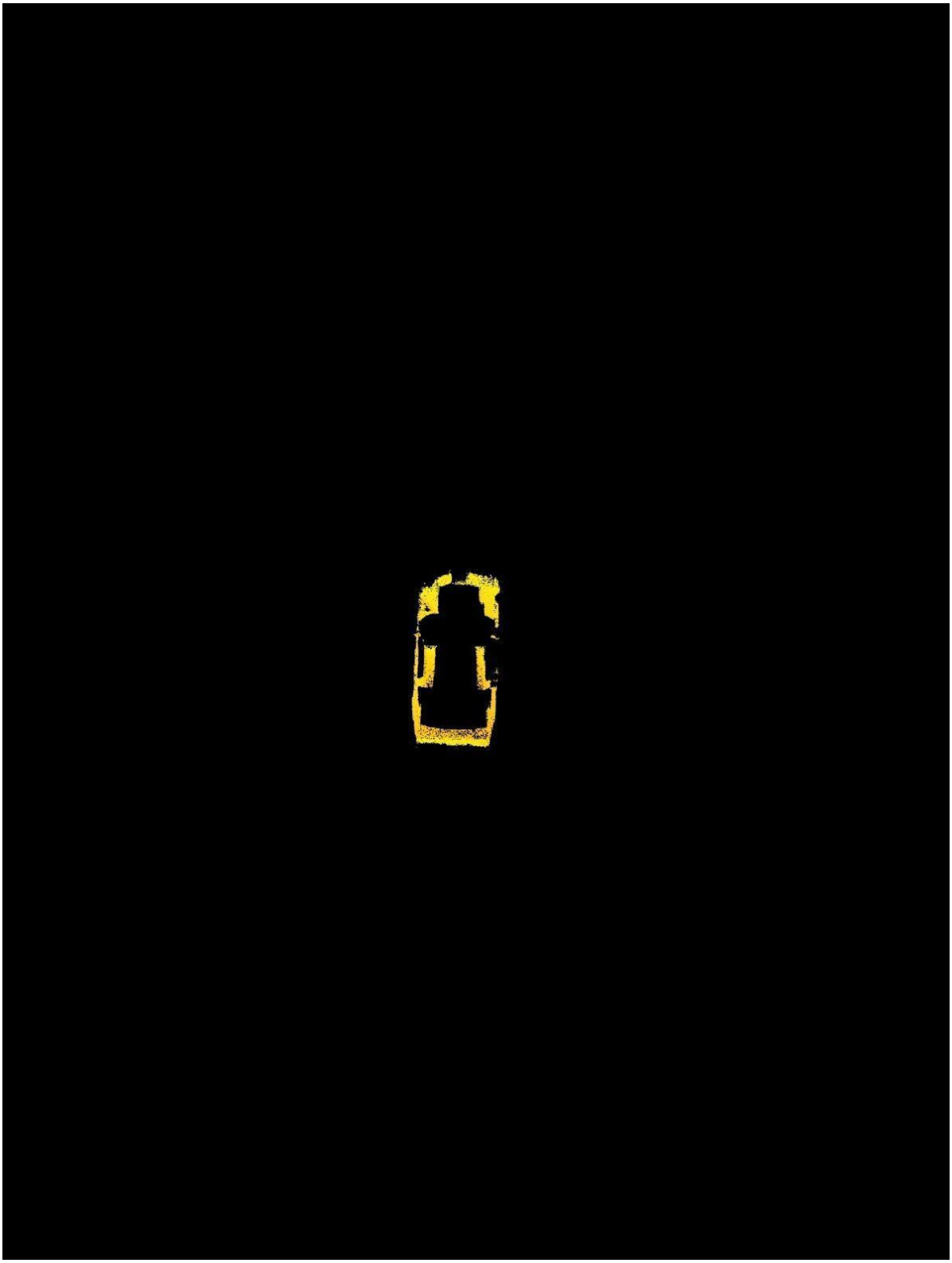
Output: A direction signal for the chasing car.

Description: Given multiple points of the trajectories of both cars, IPC will process each picture twice (once for the running car and once for the chasing car).



Firstly it filters out the background, leaving only the pixels with car's color in it.

Secondly it filters out the noise.



Here we assume the largest cluster of the pixels is the target car. And we filter out the minor ones. For the actual implementation, we will use obstacles with completely different colors such that noise pixels are less than 1% (or even zero), which can be easily determined by the distance from the center coordinate and simply removed.

Thirdly, it estimates the center coordinate of the car. Since the speed of light is much faster than the cars, we are going to assume there's no movement for the cars between the time that the scene is taken, and the time that the camera receives the picture. We also ignore the error caused by the different distances from cars to the camera. This is because we are only going to send out an estimated direction signal and we did not consider how to dodge the obstacles (this is done by the microcontroller and the sensors on the car). Even an error within 20 degrees should be fine since microcontroller needs to dodge the obstacles either way.

Lastly, with multiple center coordinates and the time difference (1/30s for each two pictures), we can estimate the direction and the average speed of the cars. IPC will send a directional signal which tells the chasing car that, with such a speed, how many degrees (0 ~ 360) should the chasing car turns counterclockwise in order to catch the running car. IPC will always compute the average of the last five to ten points the camera took, depending on the actual behavior of the car and the wireless device. Note that we do not really need to know the actual speed of the cars. We only need to know the relative speed of the cars. Thus, we only need to use pixels as the distance unit.

Detailed calculation: Given multiple coordinates of both cars, we can find out the relative speed of both cars. Define the running car's speed as V_r, and chasing car's speed as V_c. Using pixels as distance unit, we can have an (almost) linear trajectory.

Define the last several center coordinates of the running car as (X_{r1}, Y_{r1}), (X_{r2}, Y_{r2}), (X_{r3}, Y_{r3}). Define the last several center coordinates of the chasing car as (X_{c1}, Y_{c1}), (X_{c2}, Y_{c2}), (X_{c3}, Y_{c3}). We can find out the approximate average slope$K_r \approx \frac{Y_{r1} - Y_{r2}}{X_{r1} - X_{r2}} \approx \frac{Y_{r3} - Y_{r2}}{X_{r3} - X_{r2}}$ and K_c by the corresponding coordinates. We now have two equations with two unknowns: impact coordinates X and Y:

1. $\frac{Y - Y_{r3}}{X - X_{r3}} = K_r \quad \rightarrow$ The impact point should be on the running car's trajectory. Use last detected coordinate (X_{r3}, Y_{r3}) as the reference.

2. $\Delta t = \frac{\sqrt{(X - X_{r3})^2 + (Y - Y_{r3})^2}}{V_r} = \frac{\sqrt{(X - X_{c3})^2 + (Y - Y_{r3})^2}}{V_c} \rightarrow$ The time used by both cars to travel to the impact point should be the same. Here we assume the chasing car does not slow down when making turns. We can modify the details when simulating it.

After getting the coordinates(X, Y), we should be able to calculate the new direction that the chasing car should go $K_c' = \frac{Y - Y_{c3}}{X - X_{c3}}$. With the original direction the chasing car is going, we can find out the turning angle between these two directions. $\Delta\alpha = \arctan(K_c') - \arctan(K_c)$. See appendix for detailed graphical explanation.

Here the IPC is an actual computer connected to the camera. This is because we found most cheap boards do not have enough memory (usually 256KB) to store even one image, let alone to process multiple of them every second. And the ones that have such large memory are generally above $1000. We are using windows 7 system and openCV libraries to do the image processing. Language is C++. We also attached a simple code snippet we used for the sample images, using easyBMP library.

2.3.3 Communication Module

Input: Direction data from Image Process Component

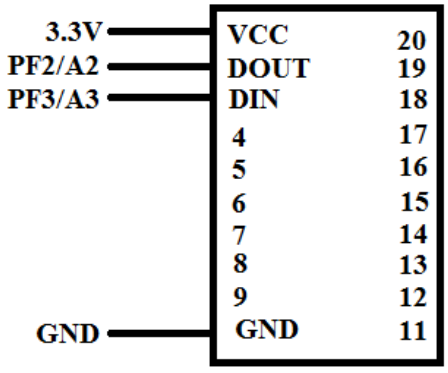
Enable signal from Microcontroller

Output: Direction data to Microcontroller

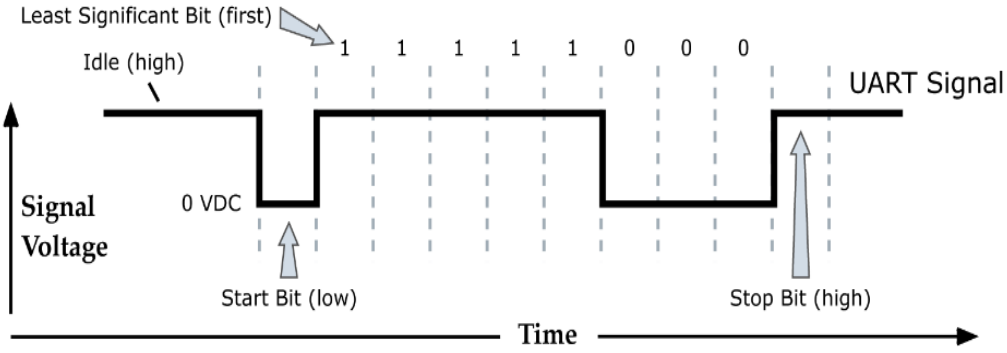
Enable signal to Image Process Unit

Description: For communication module, we use XBee®DigiMesh® 2.4. It is responsible of the wireless connection between the MCU and the IPC. It transmits the serial data generated by the IPC and the signal by MCU. The module should hibernate when the MCU is in sensor-based detection mode. And it can be signaled to activate by the MCU on demand. In camera-based detection mode, it will keep transmitting direction signal that processed by the IPC until the MCU puts it into sleep again. The XBee board directly connects to the MCU, and its power also comes from MCU. It receives an one-byte signal transmitted from the laptop and outputs the data to MCU through port pin2/DOUT. And port DIN receives the same type of signal from MCU and sends to the IPC. For the XBee board that connects to laptop, we use the USB interface board that comes along with XBee.

Pin Configuration for XBee on MCU



8-bits signal waveform that transmitted by XBee



2.4 Chasing Car Component

2.4.1 Ultrasonic Sensors

Input: The information detected by ultrasonic wave

Output: Digital output for detecting the presence of objects

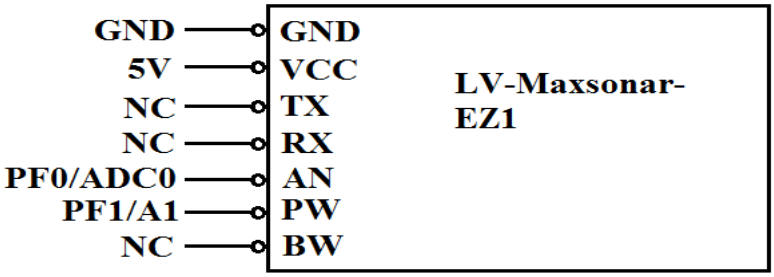
Description: We use LV-Maxsonar-EZ1 as the ultrasonic sensor. The ultrasonic sensor is used to measure the distance of objects in front of the chasing car. The sensor detects the object in its area of view and read the distance. It then converts the distance in length into three different types of outputs through the pin AN. The output we are using is the voltage output with a proportion of Vcc/512 V/inch. On condition of 5V Vcc, the output voltage is in a range from 80 mV to 2480 mV. This analog data will be sent to MCU's A/D converter, which will determine the distance and relative velocity by the equation

$$\text{distance} = \frac{\text{Vcc}}{512} \times 0.0254 \text{ m}$$

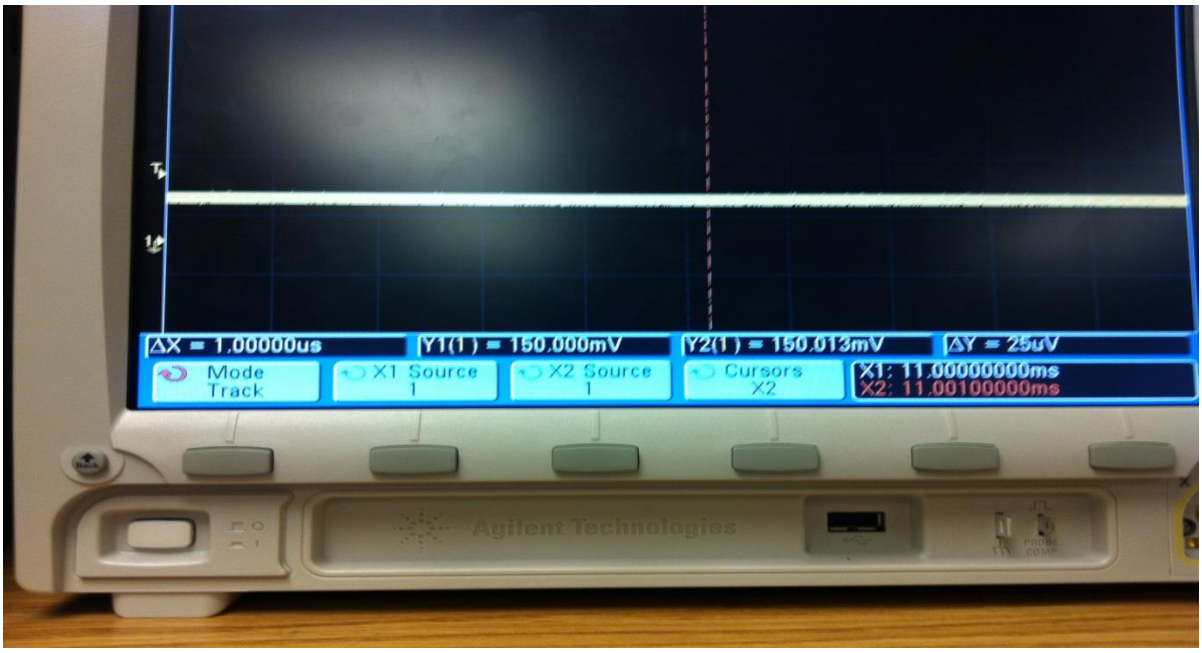
In the simulation, we put the object (our hands) at a distance of 20 inches and 15 inches. And the sensor successfully output a voltage of 199.75 mV and 150.00 mV. The sensors will be put on a

daughterboard of microcontroller with an input voltage of 5V (see datasheet). And its PW pin connected to the microcontroller to enable/disable the sensor. When we use multiple sensors, we decide to enable each sensor one by one in order to prevent the interference among the sensors. Each sensor takes 50 ms to take and receive a reading. So with 5 sensors in series, it takes about 0.25 sec to finish one cycle of detection. The sensors will output waves at 20,000Hz with a detection range of more than 20 feet. Further tests are needed to determine the accurate positions of sensors in order to guarantee the field of view has at least 120 degree angle at front.

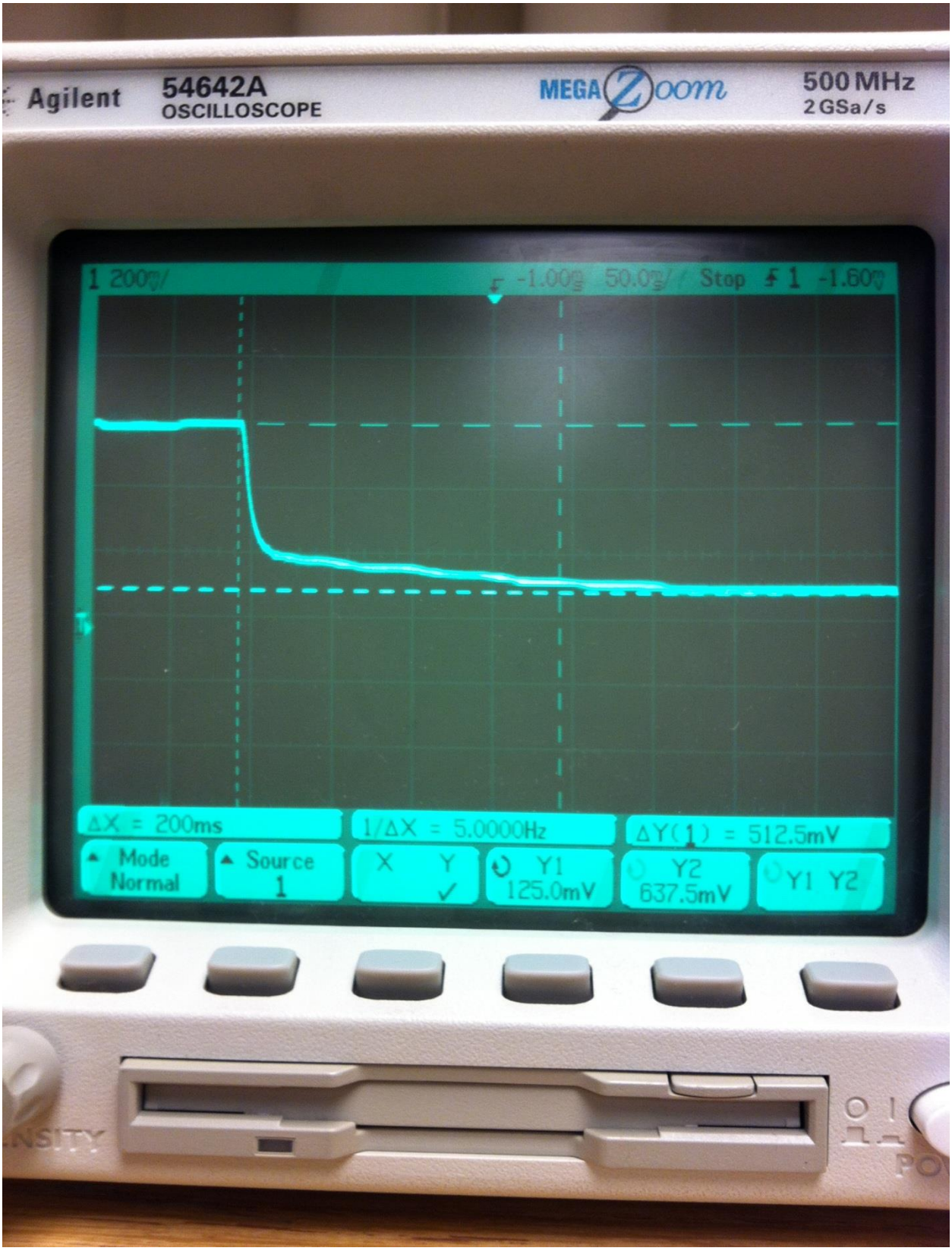
Pin configuration of Ultrasonic Sensors on MCU



Simulation with object at a 15-inch distance



Sensor voltage drop when an object comes in with a distance of 12-inch



2.4.2 Motor control module

Input: Direction signal from MCU

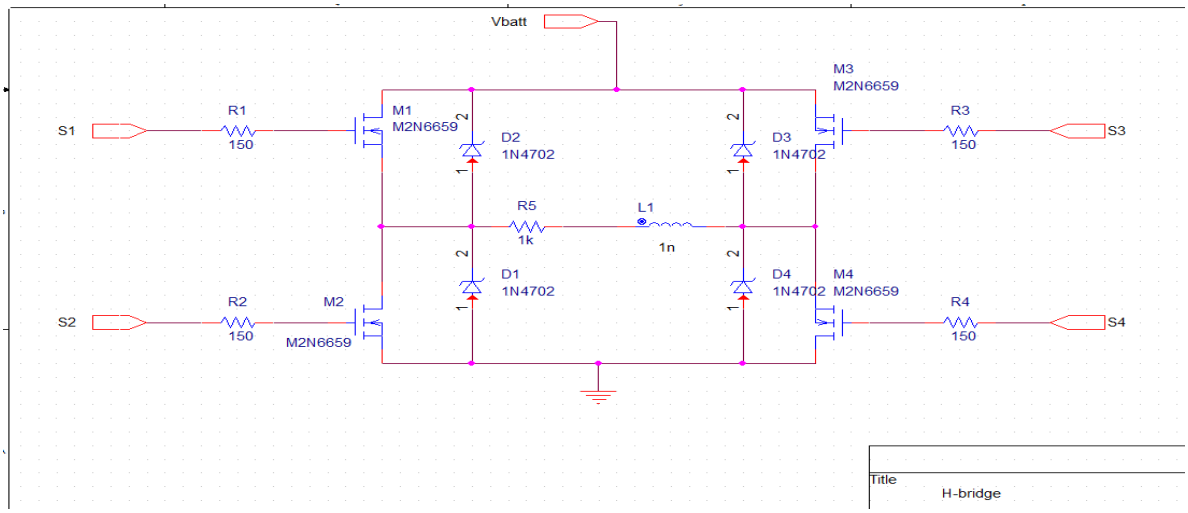
Output: The actual movement and turning of the chasing car

Description: This module is what physically drives the chasing car. We will preserve the physical turning mechanism of the toy car. In addition we will add an input signal from MCU. It uses a H-bridge circuit to control the motor and turns the front wheels to turn left or right. It turns right when it receives 1001 signal and turns in the opposite direction when it receives 0110. It keeps going straightforward when no signals received, or 0000 is received. The motor brakes when receiving 1010/0101.

Control signal

S1	S2	S3	S4	Result
1	0	0	1	Motor moves right
0	1	1	0	Motor moves left
0	0	0	0	Motor free runs
0	1	0	1	Motor brakes
1	0	1	0	Motor brakes

H-Bridge Circuit



2.4.3 Microcontroller Unit (MCU)

Input: The digital output from ultrasonic sensors,

The direction signal from communication module

Output: The movement signal to the motor control module

The enable/disable signal to the IPC

Description: The microcontroller is the key component of the chasing car. We choose Arduino Mega 2560. It collects data from sensors and communication unit, runs the pursuit-evasion algorithm and outputs the direction instruction to the motor controller. This MCU is the most important component of the whole system. It has two control modes/programs: sensor-based detection and camera-based detection.

The sensor-based detection is used when the chasing car is able to detect the running car with its onboard ultrasonic sensors. The sensors can return good distance measurements of all objects in its FOV. We guarantee that the chasing car has a fairly constant average speed that could be used in calculations. Based on the distances and the speed of the chasing car, the algorithm should be able to predict the position of all detected objects appearing in the next detection cycle. It compares the objects detected in the next cycle with the predicted locations. With this method, we should be able to separate the running car from the background including any obstacles. Then, the algorithm would turn the distance and direction of the running car into an coordinate. With three coordinates of the running car, the algorithm is able to perform a monomial basis to estimate the polynomial trajectory of the running car.

After the trajectory calculation, the chasing car can perform an aggressive pursuit algorithm. Instead of following the running car, The MCU directs the chasing car to the point that may possibly encounter the running car after a few seconds. We can also perform a lock-on

strategy so that the detection algorithm will analyze the sensor data around the trajectory at top priority.

The camera-based detection is only used when the ultrasonic sensors can no longer detect the running car. It means the running car is already out of sight. The MCU will send a signal through the Communication Unit to activate the IPC, which should be able to return a direction signal. The signal is calculated by the IPU and the calculation is in Appendix C. The MCU will follow the instruction and control the Motor Control Component until the ultrasonic sensors catch the running car again. After that, the MCU disables the IPC and switches back to sensor-based detection.

2.5 Running Car Component

Input: Signals from control panel.

Output: Movement with constant speed and various directions

Description: We are not going to modify the running car's motor too much. But in order to lower the speed to 0.1m/s - 0.3m/s, we need to split the voltage to a resistor added to the circuit supplied from the battery. Thus the motor will receive lower voltage and the speed will decrease. See the simulation in the appendix.

2.6 Power Supply

Input: 9V High Power Battery

Outlet

Output: Stable power supplies for our cars' motors, wireless devices, camera, IPC and MCU.

Description: There are two kinds of power supplies. One is for the IPC and camera, the other is for the cars' motor and MCU. Former is from 110V outlet which should have an stable power flow. Latter is a group of three 1.5V batteries(Lithium AAA 1.5V High Energy Lithium) in the running car and a 6V high power battery (Varta High Energy 9V Battery) for the motor and MCU.

For the running car, it should last more than 10 hours. Each battery has 1250 mAh while the toy car drains about 200 mA. With 3 battery, we have

$1250 \times 3 \div 200 = 18.75\text{h}$

For the chasing car, the battery have a 5 Whr rating, as we cannot estimate the actual weight of the car which greatly affects the power consume of motor, we can hardly estimate how many hours can the chasing car last running. But we can safely assume that the motor consumes no more than 250 mA. And as we separate the components from MCU, we calculate the power of microcontroller functioning at its maximum operation frequency without output any power. So, we are expecting the maximum power：

Motor: $250\text{mA} \times 1.5\text{V} \times 3 \times 2 = 2.25\text{W}$

Sensors: $20\text{mA} \times 5\text{V} \times 5 = 0.5\text{W}$

Communication: $50\text{mA} \times 3.3\text{V} + 45\text{mA} \times 3.3\text{V} = 0.3135\text{W}$

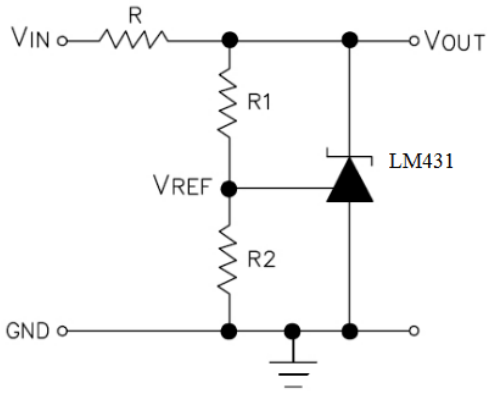
Microcontroller: $4\text{mA} \times 9\text{V} = 0.036\text{W}$

The maximum power consumed by the chasing car is 3.0095W. With a battery of 9V, a capacity of 1.2Ah, the chasing car should be able to run more than:

$1.2\text{Ah} \times 9\text{V} \div 3.0095\text{W} = 3.6\text{h}$

The 9V battery is the power source for both MCU and motor. To connect the MCU, it first goes through a shunt regulator to keep the voltage stable.

Voltage stable circuit



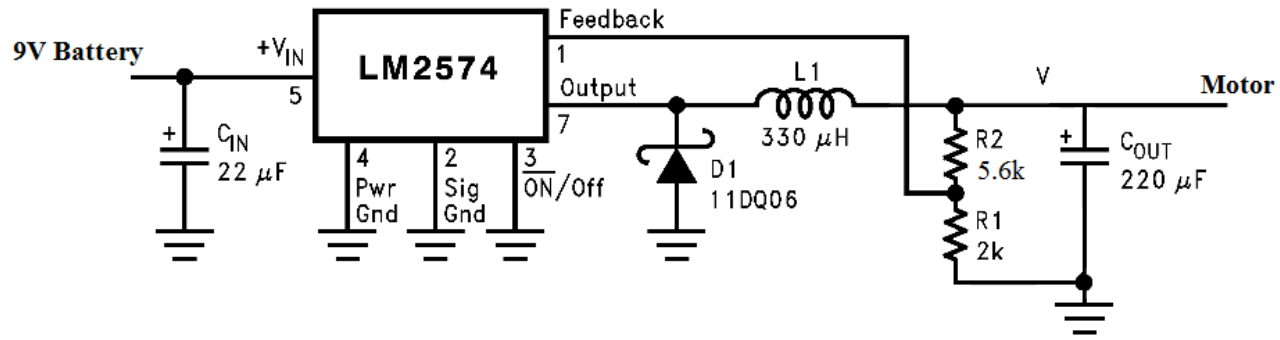
We use a step-down voltage converter scale the voltage to 4.5V so that it fits the requirements of our motor. We use a LM2574 Buck converter to do the scaling. As the voltage is calculated by the equation, we set the resistors as 2k and 5.6 which will output a voltage of 4.67V.

Step-down/buck converter with LM2574

$$V_{OUT} = V_{REF}\left(1 + \frac{R_2}{R_1}\right)$$

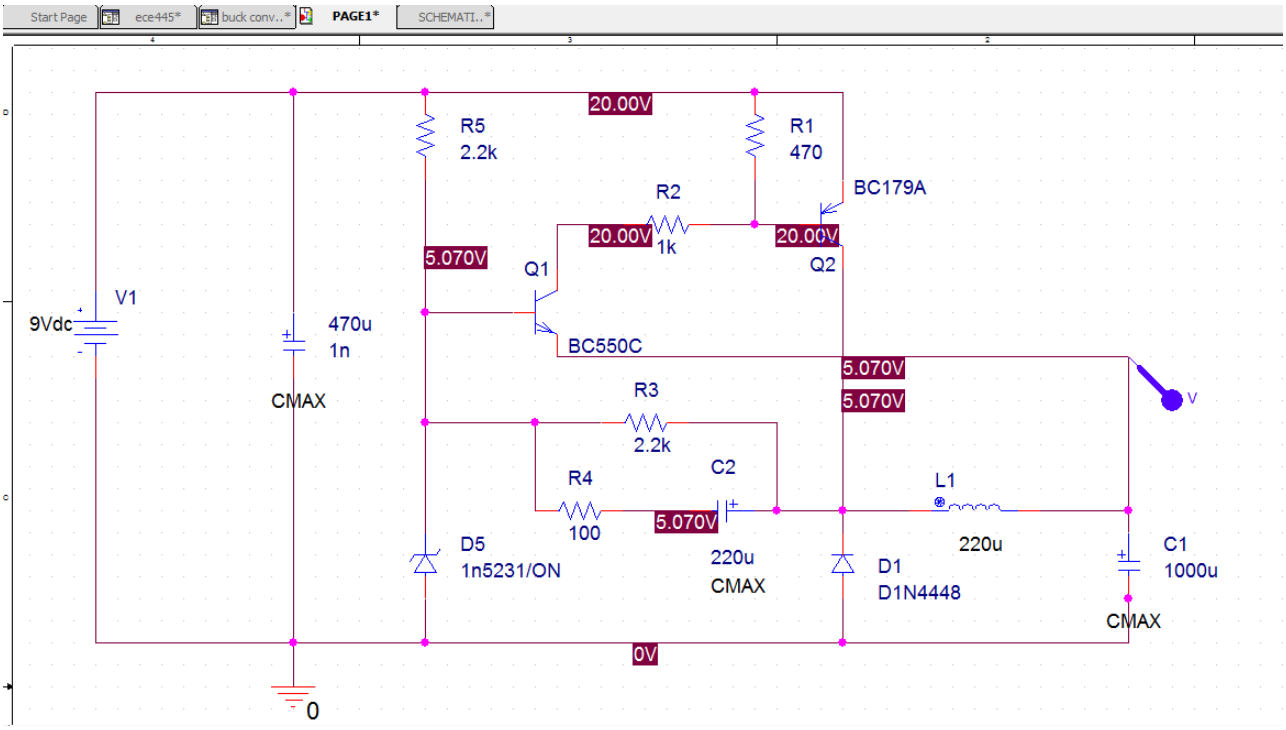
$$R_2 = R_1\left(\frac{V_{OUT}}{V_{REF}} - 1\right)$$

where $V_{REF} = 1.23\text{V}$,
R1 between 1k & 5k.

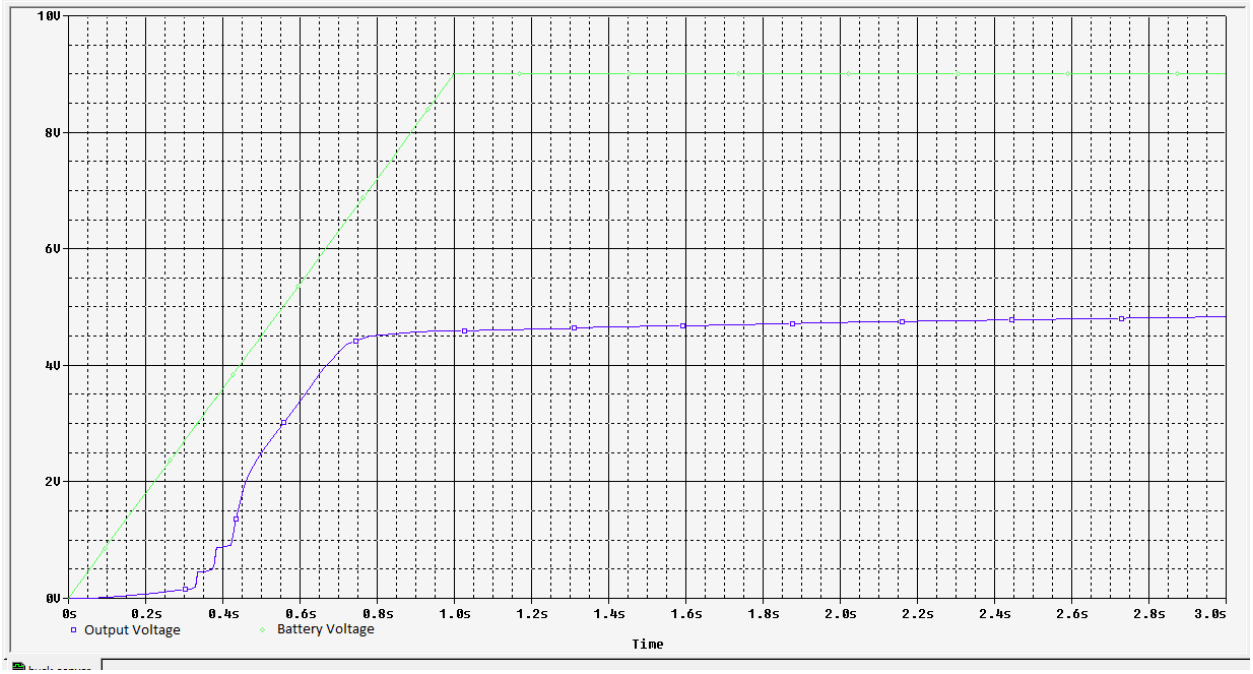


In case we fail to get a LM2574 chip, we can use an alternative circuit that has a not too much less efficiency but much simpler. See the simulation below.

Step-down/buck converter without LM2574



Circuit output simulation



The power for a typical computer is about 100W. And the power for the camera is 1W max, 0.9W typical, and 0.4W when sleep. The XBee connecting to IPC is also about 0.3135W. The running car's power is 0.2A * 4.5V = 0.9W

So the total power of every component in our project is 100 + 0.3135 + 0.9 + 3.0095 = 104.223W

3. Requirements & Verifications

3.1 Testing procedures

Requirements	Verifications
Camera The camera outputs pictures to the IPC with a frequency of 7.5Hz.	
1. Power Supply: Guarantee the input voltage is in its functioning range, which is 8 to 30 VDC.	1. Test the input using multimeter. If the wire cannot guarantee the voltage input, we need to order another one.
2. Dimension: Each picture is of dimension 640 * 480.	2. We can test this by printing out the outputs of corresponding height and width functions within openCV library. Usually this should be constantly true, but we will still test it at least 100 times.
3. Frequency: Each second it should output 7.5 pictures each second. If the frequency has an error frame rate of 2 or more, we cannot guarantee the accuracy of the direction signal anymore.	3. We will have a counter variable for the quantity of input pictures each second. Computer should print out a warning message if the counter is not within 7.5 $\pm$ 1.5 pictures. When the rate is not within7.5 $\pm$ 2.5, IPC should send a special signal to let the microcontroller know that microcontroller should not use any new data transmitted from the IPC. Under this circumstance, microcontroller will try to search the running car with its sensors, and the latest direction signal. IPC, on the other hand, should reboot the camera and check its rate counter. IPC will send another special signal to microcontroller and if at this point the microcontroller still does not find the running car using the sensors, it should start to use the data from IPC from this point.
4. Range: a. The height from the ground to the camera is about 3$\pm$0.1 meters.	4. a. Measure it with the measuring tape. For the environment, we will use the floor of a building so that we have a stable lighting environment, as well as a level ground. Thus we can guarantee the angle from the camera toward the ground is almost vertical.

b. With this height, we test the real radius of range of the camera. Make sure neither car goes outside the range.	b. Mark the edges on the ground by checking the image of the camera. Put multiple blocks in a circle such that neither car could get out.
Image Processing Component The IPC should resolve the center locations of the two cars. And based on multiple coordinates, it continuously sends direction signal to the microcontroller for the chasing car to turn. 1. Power Supply: Since IPC is a computer, we always assume it functions properly given power from the outlet. Nothing will be checked. If the computer is not working well, we can tell by its function.	
2. Prerequisite: Each picture always contains both cars.	2. As mentioned above, we will build a circle of obstacles such that no car can get outside. Since the ground is level, we should not worry about the effect of the angle. Also, we will use sponges as the obstacles. They have a very distinctive colors and soft enough so the car will not be damaged. The heights of the sponges are lower than the cars, so nothing should be able to block the site of the cars.
3. Transmission: Guarantee the connection between IPC and the camera is 100% successful.	3. The supported cable is a 2m firewire thin cable connecting both components. We already tried the connection over 20 times and it did not fail even once. But this is essential in order to get the signals from IPC. Computer should have logs when it's failing to transmit data. We will print the log on the screen when such incidents happen. If it fails at least once, we should try to order another cable online.
4. Filtering: For each colorful picture of the car, IPC should be able to recognize a cluster of points with the color of the car. a. Make sure 95+% of the pixels of the background are filtered out. b. Make sure 95+% of the pixels of the car are preserved.	4. a and b. We will have two base pixels which are almost the same color as the both cars. Of course we will paint two cars with completely different colors such that it is very clear to separate them. Specifically, we will calculate the difference between each pixel in the picture and the base

	pixel. The difference of two pixels is defined as the sum of the differences of all red, blue, green components. $\text{Diff} = |\text{red}_{\text{this}} - \text{red}_{\text{base}}| + |\text{green}_{\text{this}} - \text{green}_{\text{base}}| + |\text{blue}_{\text{this}} - \text{blue}_{\text{base}}|$ We categorize all the pixels into two groups depending on its difference: within the criterion and outside the criterion. We convert the outside ones into a completely black pixel. This is just like marking it outside the consideration/calculation. We preserve the ones inside the criterion with its original pixel and go to the next step of analysis. The specific criterion depends on the actual light condition, as well as the previous experience. By doing this, we should be able to filter out most of the background pixels and preserve the car's pixels. We will test it at least 50 times with different lighting environments.
5. <u>Recognition:</u> Only the car's pixels should be preserved. All the outside pixels should be filtered out (at least all the ones with a distance greater than the length of the car). a. All the dummy points should be recognized and filtered out. b. 99+% of the real points should be preserved.	5. a and b. In the experiment, we will minimize the conflicts between background and the cars. Ideally, all the points recognized from last step should all be real points. But with the affection of lighting and all other possible side effects, it is still possible that certain extreme points are preserved from last step. And they will affect the calculation of the center coordinate catastrophically. Thus, we should have another filtering algorithm to filter out the potential dummy points with the same color of the car. Here we definitely need to assume that there's a main cluster of points and that cluster of points should be the car and real points. It does not make sense if there are even more dummy points than the real points. We should observe the results of the tests from previous section to guarantee that among all preserved points, 95+% should be real points. If not, we need to shrink the criterion until it fits. The second assumption is that dummy points are mostly far away from the main cluster. Thus, we can group the points into different sets. Algorithm is as follows: Start with a bunch of empty sets. Define the distance of the two points

	(X_1, Y_1) and (X_2, Y_2) to be Distance $= \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$ Define a distance criterion 'D'. Iterate through all the preserved points, for each point do the following: If there exists an previously iterated point whose distance from the current point is within the criterion 'D' then: put current point into that set. else: put current point into an new empty set. After the iteration, find out the size of each set. Get the set with the maximum number of points (this set should contain the real points). Mark all points that are not in this set to be completely black. Repeat the above algorithm with a lower distance criterion 'D' until the result is almost ideal by observation. We will test this algorithm more than 50 times and guarantee it meets the requirement. We will use the best 'D' when doing further experiments.
6. Calculations: The IPC will output the directional signal. a. Given a processed picture, IPC should calculate the center coordinates for each processed picture. This is calculated by the computer and should not have any error. The error between the calculation and the reality should be within 1cm.	6. a. Now that we have an accurate set of car's pixels $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, according to their coordinates, we can just output the estimated center coordinates (X, Y) by the functions $X = \frac{\sum_{i=1}^{n} X_i}{n}$ and $Y = \frac{\sum_{i=1}^{n} Y_i}{n}$ We can test it by looking at of the output coordinate and see if it is reasonable. We will test for at least 50 times.
b. With multiple coordinates IPC should be able to find out the speed and direction. The error between the calculation and the reality should be within 10%.	b. Now that we have a series of car's center coordinates $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, We should be able to find out the average velocity. Say if we use the last five coordinates, the velocity will be the average of four velocity vectors: $v_i = \frac{Y_i - Y_{i-1}}{X_i - X_{i-1}}$. The average function is the same as above. Except we have to separate the angle and the speed. We will see the output

	as a line on a 2D dimension indicating the angle and speed, and compare it with reality. The accuracy should be within $\pm 10\%$ of the original. We do not and should not worry about this too much because due to a series of potential errors, we cannot guarantee a relatively high accuracy. But this does not matter too much because we only need a general direction for the chasing car. After it spots the running car, it will switch back to sensor mode. We will test this by over 200 times.
c. With the speed, direction, and the latest coordinate, we should be able to predict the impact point. The error between calculation and the reality should be within 0.2m.	c. As previously explained in the design of IPC, we have two equations and two unknowns. We can find the impact point using those equations. We should compare it with the reality and the accuracy should be within a radius of 20 cm. We can test it over 100 times. Without enabling the sensors, we disable the camera at some point of its chasing. Let both cars run in its original direction and check the impact point.
d. With the impact point estimation, we should be able to get the turning angle for the chasing car. The error between calculation and the reality should be within 0.2m.	d. Use the equation mentioned in the design part, we can find out the turning angle using arctan function. This should be tested along with part c. Again we will test it more than 100 times. The accuracy is within 20 cm distance of the real impact location.
Communication Module It is used by the IPC to send direction data to the microcontroller, and used by the microcontroller to send signals to IPC to switch to mode on/off.	
1. <u>Power Supply</u>: Guarantee the voltage input is stable at 3.3 ± 0.1V. a. Guarantee the power voltage supplied by IPC is 3.3 ± 0.1V. b. Guarantee the power voltage supplied by MCU is 3.3 ± 0.1V	1. a. The voltage converter inside XBee will convert USB voltage to 3.3 ± 0.1V automatically. If it does not work well, we should reorder one. b. Since MCU and XBee are physically connected. We can test the voltage supply for XBee using multimeter. If microcontroller functions well, the 3.3V port should output a stable voltage. If the XBee does not have power, we should put checking the functionality of MCU into consideration.

2. Connection: Guarantee the connection between the component and XBee. a. Make 100% successful connection between the IPC and XBee b. Make 100% successful connection between the MCU and Xbee	2. a and b. If the power flows in properly, the connection should also be fine. There's an LED on XBee, indicating if it is functioning properly. If abnormal circumstance happens, we should see if we can fix it. If not, we need a new XBee.
3 Communication: Module has 99.9% accuracy of transmitting data. a. Transmission from IPC to MCU is 99.9% successful. b. Transmission from MCU to IPC is 99.9% successful.	3. a and b. A standard XBee can send 250 kilobits every second. We are sending signals in a rate of 8 bits a time. Technically, we can send 31250 samples every second. Since we only need to send tens signals each second, we can send multiple times to guarantee the transmission accuracy. We will send each data 5 times. The component receiving it will compare the 5 results. If one of them is different with the other four, we still consider it as a success. If two or more results are different, we consider it as a failure and output a failure signal. For IPC it prints the wrong data on the screen. For MCU it outputs a failure-transmission signal to IPC and let IPC print the last result out. We will not consider a case of two consecutive failures (which sending the special signal fails, too) because we guarantee a 99.9% success. We can test this more than 10000 times and it will not take too much time due to the high efficiency of XBee.
4. Distance: The maximum distance between MCU and the laptop is more than 10m indoor without big disturb.	4. The distance between the MCU and laptop will be assured to be $\geqslant$10 meters by using measuring tape. The clearance of obstacles will be assured with visual check. Send multiple signals and test the results as above. Make sure the success rate is more than 99%. We can test 1000 times.
5. Transmission Rate: Make sure more than 10000 signals can be transmitted every second.	5. Though not necessary, we can still test this. Send signals in both directions 10000 times each second. Check the accuracy and the number of signals we received. Print out the result on the screen if necessary.

Ultrasonic Sensor Unit The ultrasonic sensor should be able to detect the distance of objects that in front of the chasing car. 1. Power Supply: Guarantee the voltage input is stable at 5 ±0.1V.	1. Since we use a daughter board for the sensor, we first need to use multimeter to guarantee that circuit on the PCB is connected correctly. Then we can test the voltage supply for sensor using multimeter. If microcontroller functions well, the 5V port should output a stable voltage. If the sensor does not have power, we should put checking the functionality of MCU into consideration.
2. Detection Range: The ultrasonic sensor should satisfy range requirements. a. The sensor should never detect ground as an object.	2. a. Mount the sensor on the car. With no object in front of the car, the sensor should return the longest reading. Adjust the car, the height of sensor and the direction of the sensor so that the sensor will not be able to detect ground surface. Now put an object in front of the car, it should be able to read the distance of the object. Now put away the object again, the received distance should go back to the longest reading again. Record the height and the angle of the sensor, make sure every sensor follows this requirement.
b. The distance measured by single ultrasonic sensor should have a ±3cm error	b. Measure an object with one sensor. This test should run in 5 different distances: 30cm, 50cm, 1m, 2m, and 5m. First use ruler to make sure the distance is correct. Place the object directly in front of the sensor and take the distance reading. Then compare the two measures. After these tests have passed, we place the object that is to the left side a bit while keeping the distance the same. Do all these tests again, and compare the results. Then we place the object to the right side a bit and the tests again. The ultrasonic sensor should have readings that satisfied the requirement as long as the object is within its detection area. If these tests fail to meet the tolerance, we should consider buying more sensitive sensors since this is the lowest requirement for our project to work.

c. The ultrasonic sensor should have > 5m detection range	c. Measure an object in at least 5m distance. We put the sensor in the center. Then draw a circle with a 5m radius. Move the object along the line and record the reading. Plot the reading in graph. Check all the readings to see whether they all satisfy the requirement c.
d. The sensors have a >120 degree field of view (FOV)	d. We first test the detection width of one sensor. Use two symmetry objects to approach the sensor from both sides. These objects will be 4m away from the surface of sensor. Make them stop when the distance readings change. Then we can measure the ultrasonic wave detection width. With this width, we put sensors pointing in different direction. Then we move an object from left to right and test whether at least one of the sensors can always detect the object in the chasing car's 120 degree FOV.
3. Interference: The ultrasonic sensors should avoid interference from each other. a. The sensor should not send any ultrasonic wave 1ms after it's disabled	a. The enable/disable bit is controlled by MCU. Put two sensors with imbalanced positions to the object. We simultaneously activate two ultrasonic sensors. They should both have invalid readings due to the interference. Then at time 0, we disable one sensor and check the readings of the other active sensor. Record the time period when is the incorrect reading become correct. Mark it as time t. Then we check whether t is less than 1ms. Repeat this test 50 times with different sensors, objects and distances, the sensor disable time should not exceed 1ms. If it does exceed 1ms, we need to measure how long will it be safe the switch to another sensor. This will limit the valid reading frequencies every second. But it is always the first priority to make sure the measurements are correct.
b. The sensor will take correct distance reading within 50ms after enable signal sends.	b. We first disable the sensor. And then, at time 0, we activate it. Record the reading from 0 to 60ms. See whether the distance reading can become correct within 50ms. Do this test 50 times with different sensors, objects and

	distances, the sensor should satisfy this requirement every time. If not, we need to measure how long will it be safe to start re-reading the measurements. This will limit the valid reading frequencies every second. But it is always the first priority to make sure the measurements are correct.
Microcontroller Unit The microcontroller is the main controller on the chasing car. It receives the signal from sensors and communication unit, and it outputs correct movement instruction to the motor unit 1. Power Supply: Guarantee the voltage input is stable at 9±1V. 2. Prerequisite: We assume that the ground surface is flat, as assumed in camera part. The gaps between obstacles are larger than the length of the chasing car and the running car. The obstacle is high enough to be detected by the ultrasonic sensors, but low enough to not block the view of the camera. The number of obstacles (with gaps) in site is less than three so that MCU	1. The power source directly comes from the 9V battery. We use a voltage stable circuit to guarantee the voltage is within the range. We can test the voltage supply for MCU using multimeter. If battery functions well, the MCU should be able to have a stable voltage. If the it does not have power, then the battery is not qualified. We must replace it with a new and more powerful battery. 2. The ground surface should be flat. If it is not flat, then the sensors may incorrectly detect the ground surface as an obstacle. We already plan to test it indoor so this should not be a problem. Since ultrasonic sensor does not have the ability to detect the shape of an object, we can hardly find a way to decide whether the gap is large enough for the chasing car to go through. We will measure the gaps using measuring tape to

can handle the situation	ensure it. Also, the sensors should correctly return the distance of objects in front of the chasing car.
3. <u>Obstacles and moving object recognition</u>: It separates the running car from obstacles. a. It should record the distance of each obstacle that is in the FOV of chasing car. For each object it detects, MCU should keep track of their records until they are out of sight up to 8 detection cycles.	3. a. Based on the angles on the sensors and the speed assumption of itself, MCU can locate each object according to the distance readings. At first, its view has nothing. After it sees objects, it creates an array of empty locations to fill in. For each detection cycle it revises the updates the coordinate of that object (if it's within a tolerance of the prediction). If it is not within that tolerance, we mark that object as 'disappeared'. Keep on checking the next predicted area after 8 detection cycles (two seconds). If it still does not appear in the predicted area, delete that variable permanently. When writing the algorithm, we can test this by connecting it to the computer and simulate the results. It will output all the objects it sees with their trajectories and show it on the screen. If it appears after several cycles and disappear again, we should mark it as 'appear' and 'disappear' again. And we rerun the algorithm mentioned above recursively.
b. It predicts the new position of each object before the next detection cycle. The criterion should be within 5 cm.	b. Based on the angle of the sensor and the speed of the chasing car, we should be able to predict the object's appearing area in the next cycle. For example, assume the sensor in the very front is detecting an object with distance 1 m, and assume the running car has a speed of 40 cm/s. Further we assume the detection frequency is 4 Hz. The next appearing area for this object should be in front, with a distance of 90 cm. The tolerated area is a radius of 5 cm. We will test the accuracy with still obstacles above 100 times and adjust the tolerated radius accordingly.
c. It compares the new detection position to the predicted position.	c. Calculate the distance of the predicted coordinate with the actual coordinate use the distance equation mentioned in IPC verification.
d. It is able to recognize the running car as a new object if the distance of an object	d. If an object appears outside the appearing area, meanwhile the original object disappears,

between the detected position and the predicted position is larger than a criterion.	we count this as a "disappear-appear". If this "disappear-appear" happens consecutively, with a reasonable pattern, we recognize it as the running car and start to predict the running car's trajectory, using the last several "disappear-appear" locations. We should heavily test this as this is the core of our project. Make sure the algorithm work well and it can accurately predict the running car with a tolerance of 10 cm each cycle.
4. <u>The pursuit-evasion algorithm</u>: It calculates the trajectory of the running car. a. Based on several previous coordinates of the running car the sensors detected, it is able to predict the trajectory of the running car. The trajectory should have a tolerance within ±3 cm.	4. a. We first use Matlab to simulate the algorithm. Then we code the MCU and print its output on laptop to see whether it works. The trajectory may not accurately predict the path of running car. But it can at least tell the direction of the running car in the next two detection cycles. To test this algorithm, we will have a long detection cycle. We now decide to use a 5s cycle. The sensor first does a reading. Then we move the running car to another position. This movement should follow a fixed curve of quadratic or cubic function. After 10 detection cycles, we compare the calculated trajectory and curve function we used. They should have close slope at each detection points. We run 10 different curves. If they are not similar, then we need to come up with a better algorithm.
b. The pursuit algorithm should be able to point out a better path to chase the running car rather than following behind it. A better path means the it has shorter distance than the path of running car from the initial point to the intersection of these two paths.	b. A better path means the it has shorter distance than the path of running car from the initial point to the intersection of these two paths. We also first simulate this in Matlab. Matlab should print out our pursuit path. We compare them with the trajectory of the running car and decide whether this path is better. Then we code them in MCU. And again, we use a 5s detection cycle. At each point, the algorithm should output the predicted trajectory and the pursuit path directions. We connect these directions into a curve manually or by Matlab. Then we decide whether this path is better. We will have no obstacle in this test. We do this test for 10 times. We will try to adjust the coefficients to gain a more aggressive algorithm.

c. The evasion algorithm should be able to adjust the path so that the chasing car will not bump into the obstacles. The car should never crash into obstacles.	c. The evasion algorithm is used to avoid obstacles. As we set the gaps larger than the car itself, it should be fine. We still begin our test in Matlab. Given the pursuit direction, the evasion algorithm will adjust the direction to one that could successfully avoid the obstacle. Then we code on MCU with a tentative detection cycle which is 0.25s. We use XBee communication module to send the path direction and drive the car towards or approaches an obstacle. Then we can decide whether this algorithm works. We do this test for 10 times. If the car successfully avoids all the obstacles, then we consider this requirement passed.
5. Camera-based algorithm: The MCU will follow the order of IPC to drive the car when in the camera-base detection mode. The tolerance is 10 cm. At the same time, the sensors keep searching the running car. Once it finds the running car, MCU should disable the IPC data and goes back to sensor-based algorithm.	5. Instead of using camera and IPC, we manually test this requirement with the laptop. We give direction orders from laptop and drive the car around. If the car runs to the target within a tolerance of 10 cm, then first half of the requirement meets. Then we test the second part. We will control the running car to run around and give direction orders to the chasing car. We will add a LED on the MCU to tell if the sensors detect the running car. Our laptop should receive a disable signal when the LED lights up. There is no obstacle in this test and we will run this test 10 times. If anything goes wrong, then either the code has a bug, or the LED is built wrong.
6. Motor Control: According to the calculation of the algorithm, the MCU will send the straight, left and right turn signal to the H-bridge circuit of the Motor.	6. We code the MCU a series of direction order. Then we run the chasing car and see if follows that path within a tolerance of 10 cm. We give 10 sets of different order series. If the car runs in wrong direction in any of these tests, then we need to check our code.
Motor Control Module (Chasing car) The motor control module can control the chasing car's turning and it should provide a constant speed for the chasing car	
1. Endurance: Make sure the car can run more than 3 hours.	1. According to our calculation, the chasing car can run about 3.5 hours with the MCU in full operation. So it should run more than 3 hours with an idle MCU. Replace the battery with a

	brand new Varta High Energy 9V Battery. First make sure the voltage input is 4.5V by the multimeter. Then keep the MCU sending 0000 forever. Flip the chasing car with the roof on the ground so that the wheels are turning but the car puts still. Put it aside of me, pick a favorite movie and start to watch. Record the start time and the moment when the car starts to slow down. Check the time difference. If the time is shorter than 3 hours, we should re-measure and estimate the power dissipated in the components and come up with a better result.
2. Speed: Motor must be able to keep the running car at a speed of 0.3 m/s, and chasing car at 0.4 m/s. Both of them are within ±0.05m/s error.	2. Test the fully loaded vehicle with a fully operating MCU to ensure that the unit has the ability to run greater or equal to 0.3m/s. Visually observe if the average speed is about the same during the test. This will be taken directly from a distance over time relationship. The test should run in both running car and chasing car with 3 test distances: 5m, 10m, and 20m. If the error is too large, we should measure the power dissipated in the components. Also check if the voltage input is stable.
3. Turn: The motor control module should be able to perform a turn either left or right in the direction provided by the MCU.	3. Given the turn signal from MCU, the vehicle should turn as expected. This only tests the direction that the car turns. Then, the MCU signal the motor back to go straight. The motor should be able to turn back to go straight. Take the same test with different energy left in the battery to test the loyalty of the motor. Remember the critical point and be aware to change battery when reaching that point.

3.2 Tolerance Analysis

The most important part of the analysis is the recognition of the running car by both the sensors and the camera on the ceiling. The coordinates of the running car might be off for a few samples. We must guarantee that the error of the coordinates must be within ±1cm. As long as this requirement holds, the algorithm should be able to figure out the right track.

The delay in wireless communication should not affect our result much because the cars run in 0.3 - 0.4m/s. The distance between two modules is around 6m. It only takes 20ns for the signal to travel and less than 1ms for the communication module to react. So, the delay is not a problem.

There will be Doppler Shift Effect when chasing car uses the ultrasonic sensors. It happens when the chasing car is moving. With the formula:

$$f = \left(\frac{c + v_{\mathrm{r}}}{c + v_{\mathrm{s}}}\right) f_0$$

The speed of light is much greater than our cars' speed. The frequency should not change much. We can be sure that this shift would not affect our result.

4. Cost Analysis and Schedule

4.1 Cost Analysis

4.1.1 Labor Cost

Name	Rate	Hours	Hours * 2.5	Total ($)
Hai Chi	30/hr	20/week	500	15000
Zhe Ji	30/hr	20/week	500	15000
Total				30000

4.1.2 Parts Cost

Item Name	Unit Cost ($)	Quantity	Total Cost ($)	status
Fire-i digital camera	100	1	100	To Be Ordered
LV-Maxsensor-EZ1	27.95	5	139.75	Bought
Laptop	300	1	300	In Lab
Toy Cars	40	2	80	Bought
Arduino Mega 2560	49	1	49	Bought
Varta High Energy 9V Battery	14.03	1	14.03	To Be Ordered
XBee®DigiMesh® 2.4	25	2	50	To Be Ordered
Resistors,	0.1	35	3.5	In Lab
Capacitors	0.15	15	2.25	In Lab
LEDs	2	2	4	In Lab
Total			742.53	

4.2 Schedule and responsibilities

Week	Tasks	Members
2/5	Write Proposal	Hai
2/12	Image Processing Algorithm	Hai
	Microcontroller Algorithm	Zhe
2/19	Prepare Design Review	Hai
	Prepare Design Review	Zhe
2/26	Sensor	Hai
	Power	Zhe
3/5	Movement Module Circuit	Hai
	Movement Module Mechanical	Zhe
3/12	Prepare for Individual Progress Report	Hai
	Prepare for Individual Progress Report	Zhe
	Image Processing Threshold and Noise	Hai
	Image Processing Color Plane Extraction	Zhe
3/19	SPRING BREAK	
3/26	Communication Module Send	Hai
	Communication Module Receive	Zhe
4/2	Testing and Debugging: Image Processing	Hai
	Testing and Debugging: Pursuit-evasion Module	Zhe
4/9	Build Mechanical Chassis and Integrate Systems Together	Hai
	Final Testing	Zhe
	Ensure Completion	Zhe
4/16	Prepare Demo Presentation and Paper	Hai

4/23	Final Demo	Zhe
4/30	Final Presentation and Final Paper	Hai

5. Ethics and Safety

5.1 Ethics issues

We agree to uphold the IEEE Code of Ethics, and will address any relevant ethical concerns about our project. The ones related to our project are the followings:

3. to be honest and realistic in stating claims or estimates based on available data;

5. to improve the understanding of technology; its appropriate application, and potential consequences;

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 treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin;

9. to avoid injuring others, their property, reputation, or employment by false or malicious action;

10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

5.2 Safety issues

The power voltage of every equipment on both cars is below 12V so it does not have any potential harm to human body. But we need to be aware to properly handle the power of the computer and camera. Also we are going to solder components together so we should be aware to uphold soldering safety code (see citation).

The running car is controlled by us and it weighs less than 500 grams and the maximum speed is below 0.3m/s. It would not be able to make any kind of harm even to a baby (as long as he does not eat it). The chasing car has a weight below 1 kilogram and the maximum speed is below 0.5m/s. It also has a focus only on the running car so it will dodge anything that's not yellow and running. We promise to keep our cars away from children.

6. References

Pursuit-Evasion Algorithm

http://vp.dei.ac.in/wiki/index.php/The_Monomial_Basis

http://en.wikipedia.org/wiki/Pursuit-evasion

Arduino Mega 2560 Data Sheet

http://www.atmel.com/Images/doc2549.pdf

XBee Communication

ftp://ftp1.digi.com/support/documentation/90000991_B.pdf

Ultrasonic Sensor

http://www.maxbotix.com/documents/MB1010_Datasheet.pdf

H-bridge

http://en.wikipedia.org/wiki/H_bridge

Soldering safety

http://naples.cc.sunysb.edu/Admin/HRSForms.nsf/pub/EHSD0348/$File/EHSD0348.pdf

IEEE Code of Ethics

http://www.ieee.org/about/corporate/governance/p7-8.html

LM2574 Buck Converter

http://www.ti.com/lit/ds/symlink/lm2574.pdf

7. Appendices

A. IPC Code Snippet

```
#include <string>
#include "EasyBMP.h"
#include <cmath>
#define target "Tests/camera3.bmp"
#define out "Tests/cout3.bmp"
#define red 253
#define green 190
#define blue 5

int main ()
{
	BMP pic;
	pic.ReadFromFile(target);
	for(int w = pic.TellWidth(); w >= 0; w--)
		for(int h = pic.TellHeight(); h >= 0; h--)
			if(fabs(input(w, h) -> Red - red)
			+  fabs(input(w, h) -> Green - green)
			+  fabs(input(w, h) -> Blue - blue)  > 50)
			{
				output(w, h) -> Red = 0;
				output(w, h) -> Blue = 0;
				output(w, h) -> Green = 0;
			}
	pic.WriteToFile(out);
	return 0;
}
```

B. IPC Calculation