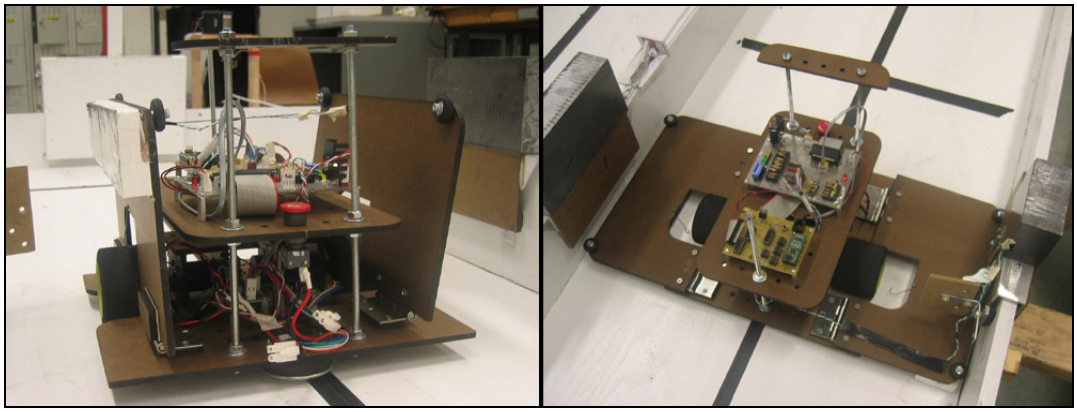
2004-2005 Winter
ME218B: Smart Product Design B
Stanford University

Robot Design and Robot War:

Teeter-Totter, Tug‘o-War



Lee Zune,

Kwon Jin Sung,

Park, Kwan Kyu,

Lee Won Young

Index

I. Design Concept

1. Define of the Game

(1) Object of Game

Hit bell ringer outside of field. : There are 2 different methods for hitting the bell ringer. First, we can move to point 1,2 or 3 to move center mass to our position to gain enough angle to hit bell. In addition we can move personal mass. In other way, we can hit the bell ringer directly. In this case we cannot use ballistic method to hit the bell ringer.

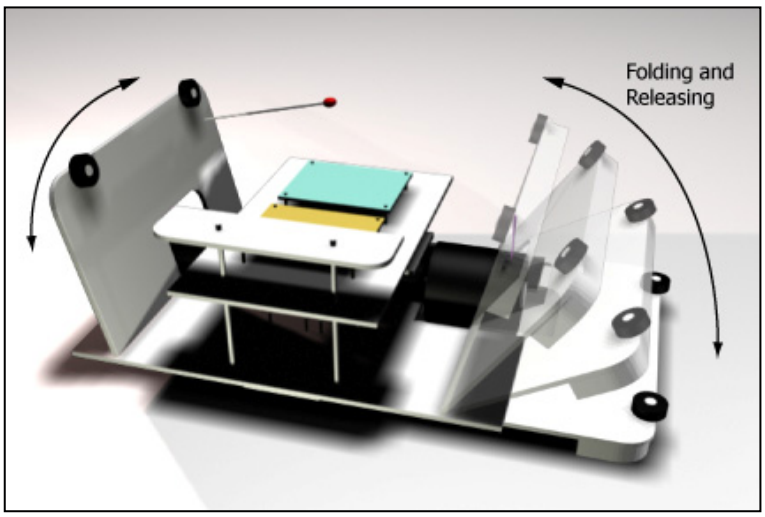
(2) Major Point

We chose to use first method to hit the bell ringer and set up out major point of design as high speed to point 3. In order to hit the bell ringer, we need to get enough angles of inclination. To get the favorable angle, we need to move center mass toward our position.(Moving personal mass is an option.) After several case studies, we found the fact that no one cannot win against the robot which can reach the point 3 at faster speed with a good strategy. In addition, we can get more flexible strategy when we get the point 3 earlier than the opposite.

(3) Team Design Concept

A. High Speed

To move reliably and rapidly, we constrain the movement only on a straight line. Thus the robot can move only forward and backward. To constrain the movement, two wings are deployed on each side of the robot. But the total width with two wings exceeds the size regulation. So to satisfy the regulation, the wings are folded before start of a game and are extended when beginning.



B. Standing Position 1,2,3

To detect proper standing point, we use line tracking sensors. We want to reject noise of signal and potential contamination of field, so we use 2 sensors to detect black lines. Only

if 2 sensors detect black line at a same time, robot recognizes black line.

C. Ability to climbing angle

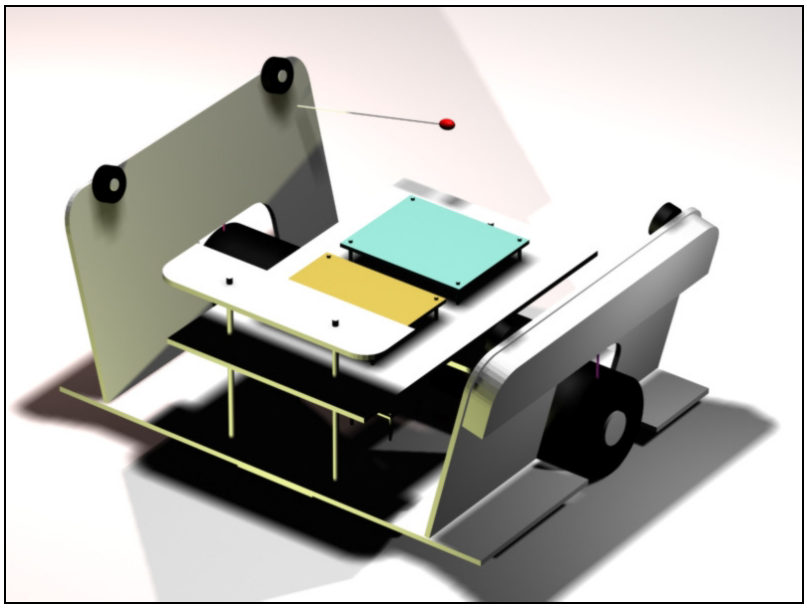
We have a trade off – Speed and Power. To get enough power we did not use gear to increase speed. After some experiment, we came to the conclusion that the power of the motor is enough. On the other side, we experienced slipping of tires. So we tried to select tires with high grip.

D. Hit Bell Ringer

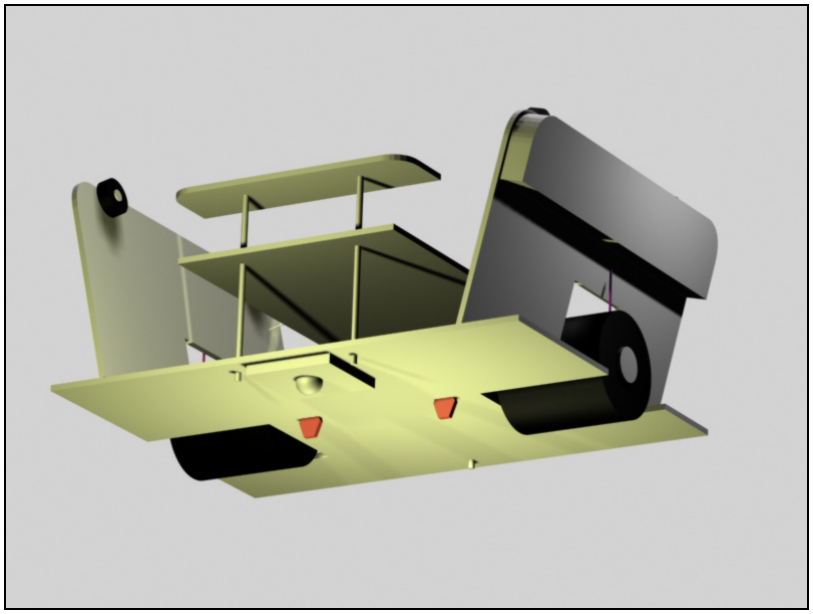
When we hit the bell ringer, we need to hit at higher position of the ringer to get enough force to press the button.

2. Design views from diverse directions

(1) Perspective views

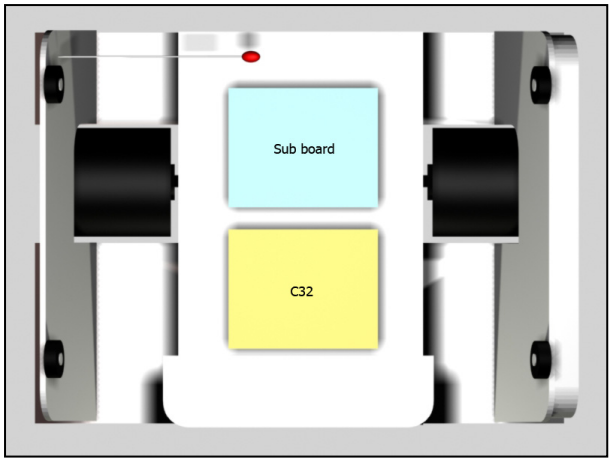


Perspective View 1

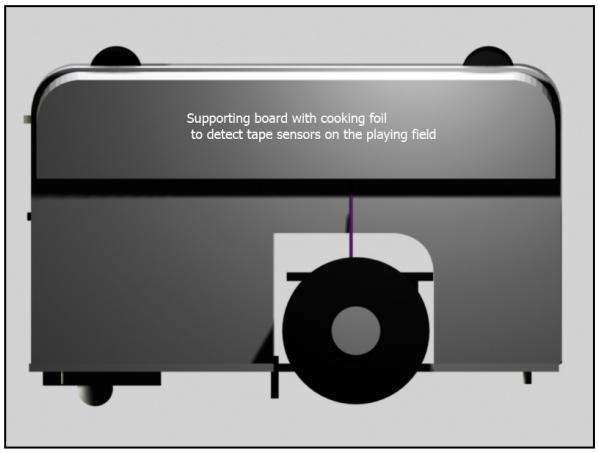


Perspective View 2

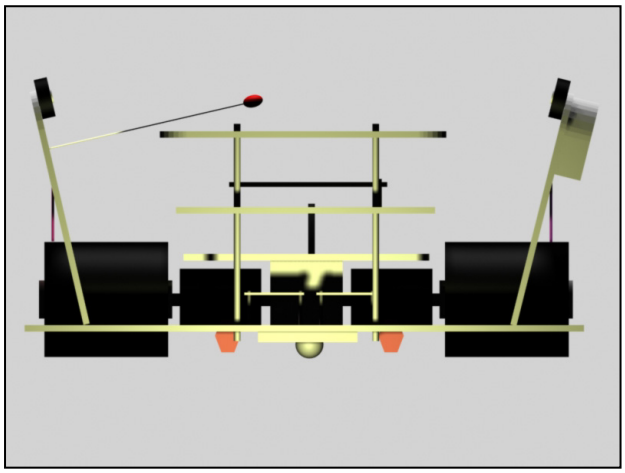
(2) Plane views



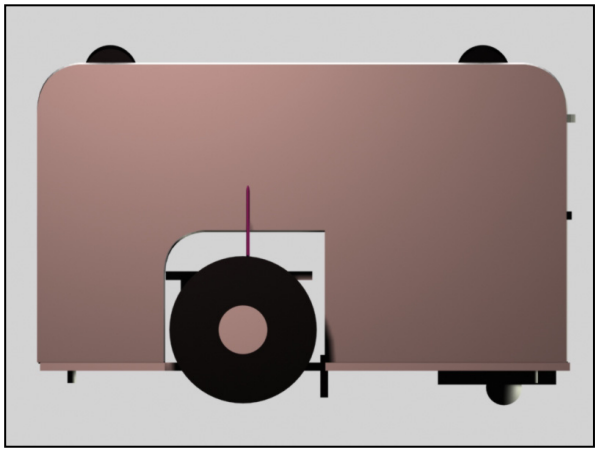
Top view



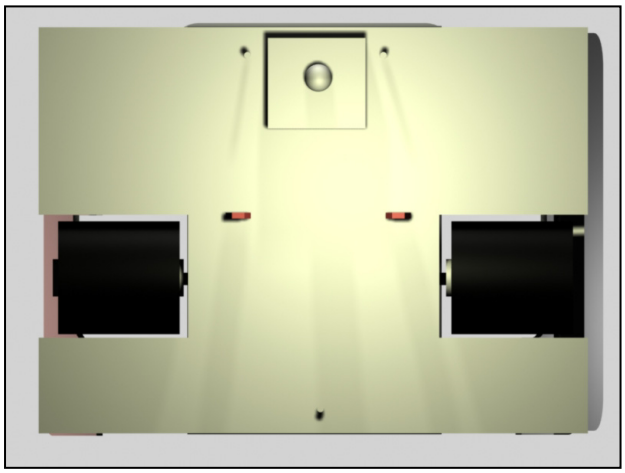
Left view



Front view

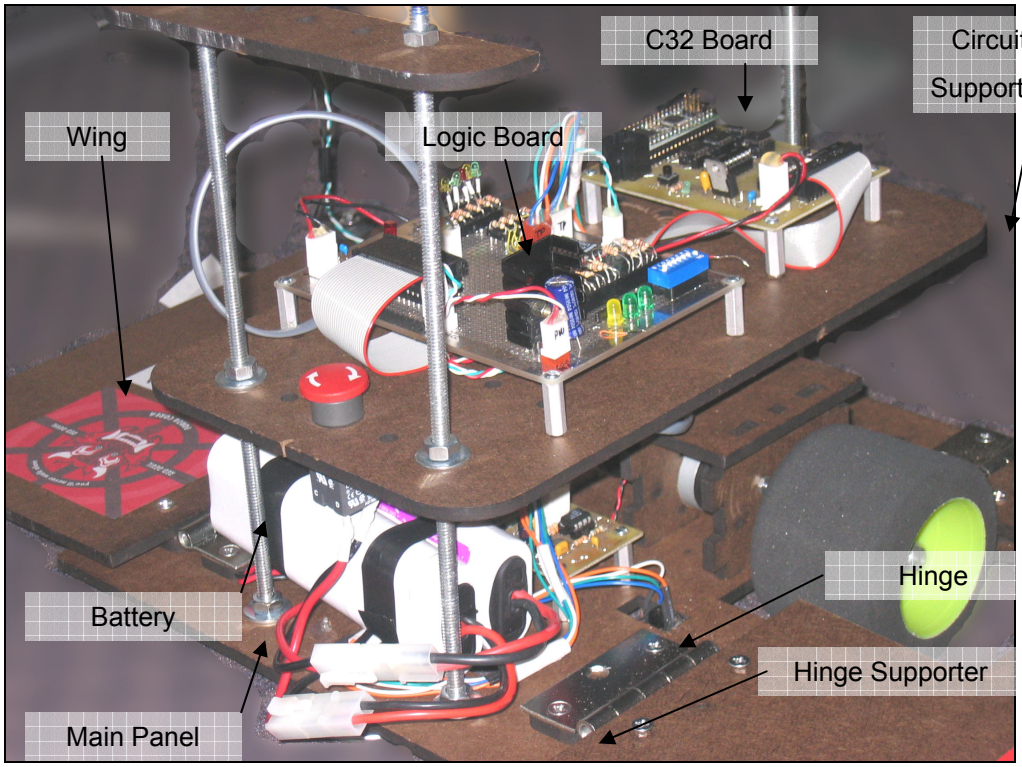


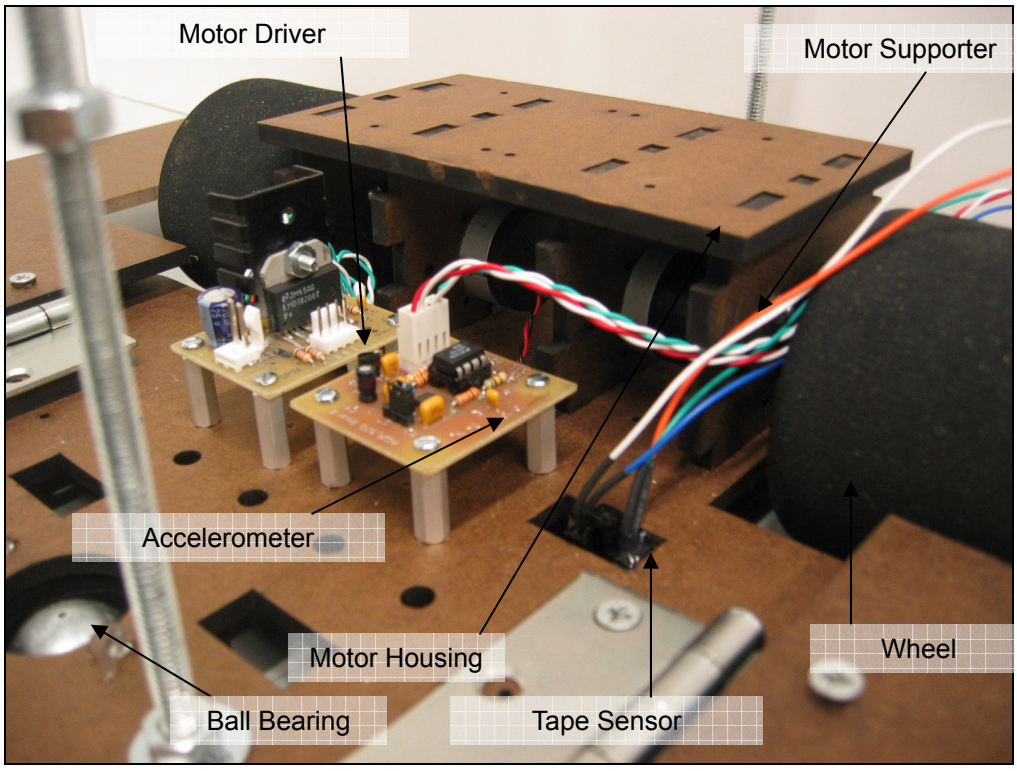
Right view



Bottom view

II. Mechanical Structure Design





1. General Design

Robot is designed to drive with 2 wheels which run in the same direction. All of the parts are located on the 12*12inch masonite panel. The hardware is designed to meet following requirements.

- Solid Structure to endure several impacts during testing and competition.
- Simple Design, easy to manufacture, easy to fix
- Light weight to get high speed.

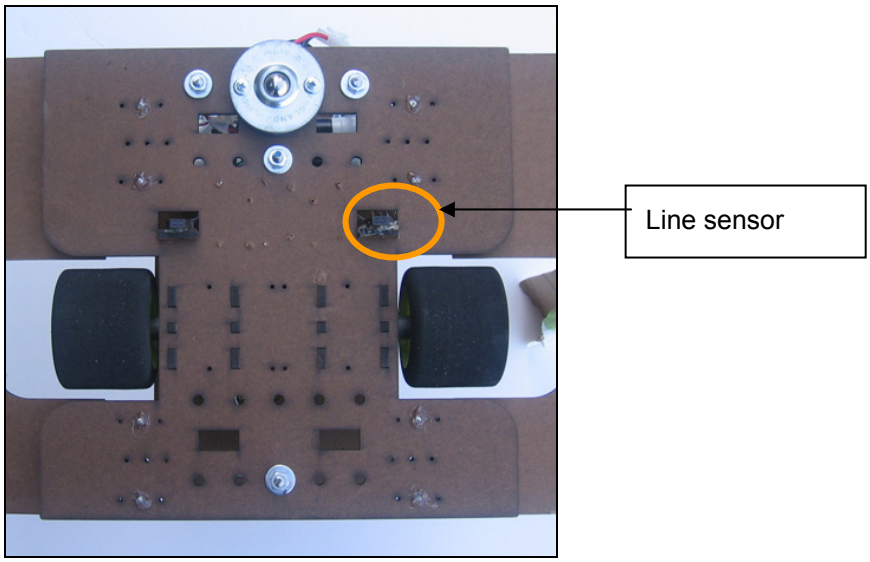
Second floor have logic board and C32 board. First floor has all other elements.

2. Functions of parts

(1) Structure

A. Main Panel

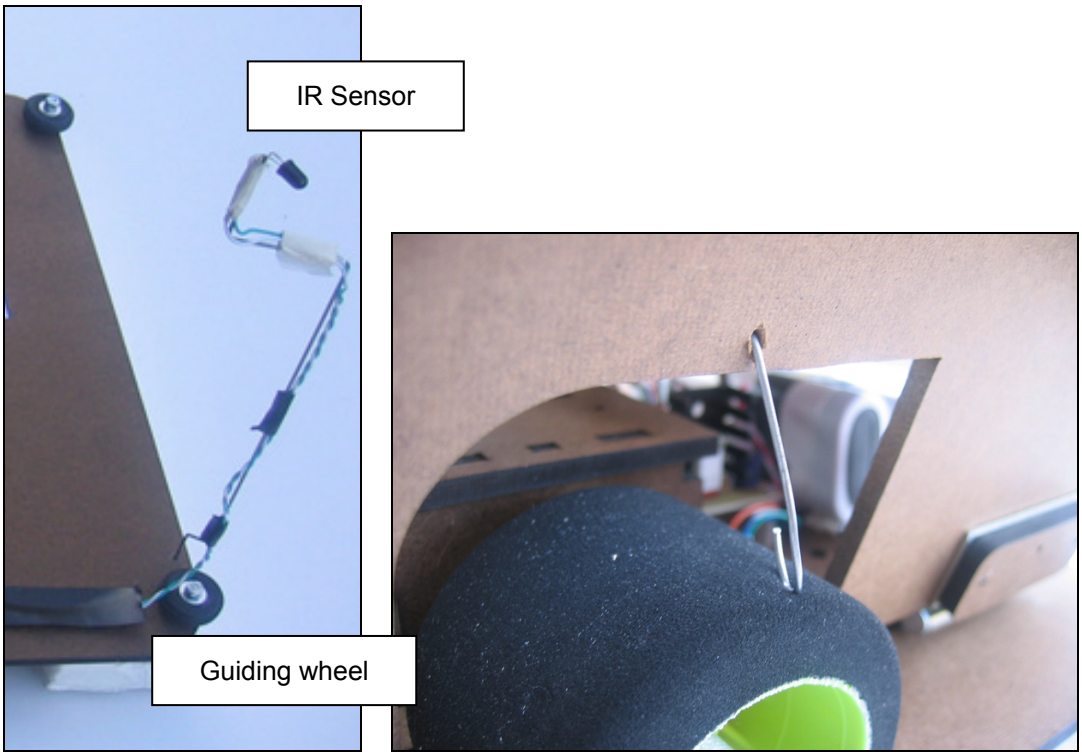
Main panel support all structure of robot. Basically it has dimension of 12”*12” trimmed for wheels. It has many holes which are used to fix motor supporters and make some space for line sensors.



B. Wing

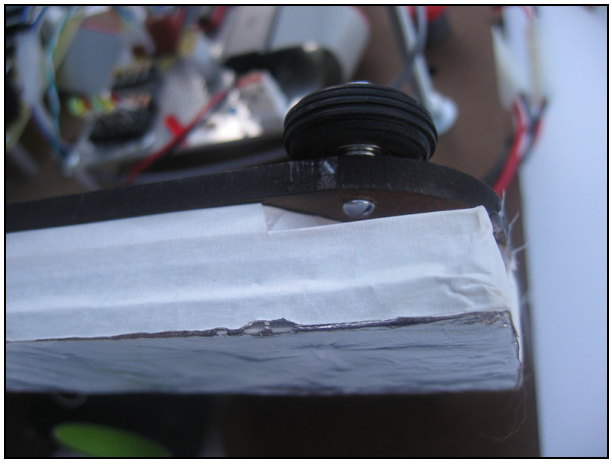
Wing is deployed at start. After deployment, dimension of robot become 12”*23” which make robot go back and forward without line tracking. In the end of wing, there is small wheel which reduce friction with wall.

On the left wing, IR sensor located which detect frequency and duty ratio of beacon.



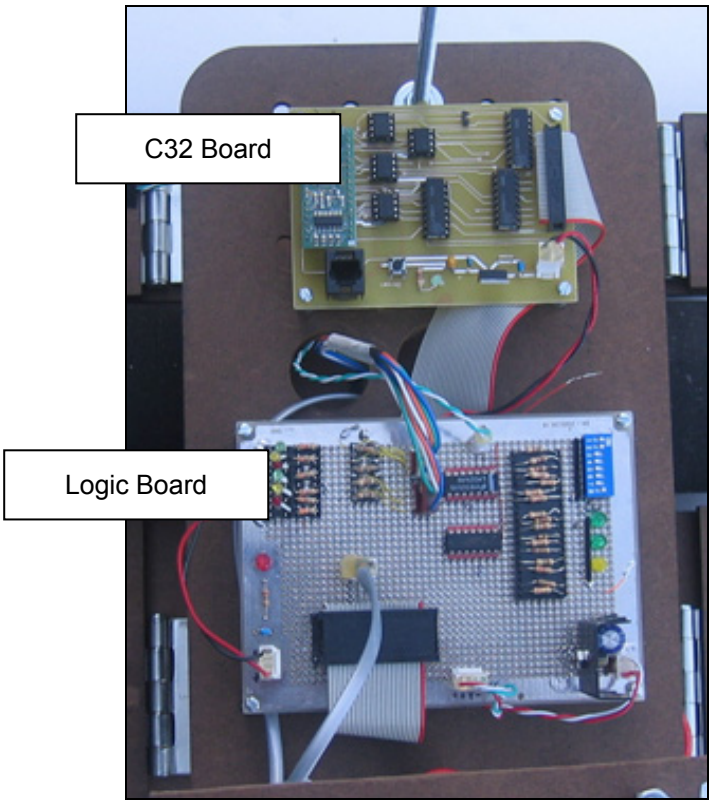
Deployment is accomplished in mechanical method. Before start, wing is stand in support of thin metal sting which is also supported by wheel. When start, metal string lose its support of wheel, and wing also lose its support of string. Because wing is slightly leaned to outside of robot, wing is deployed by gravity.

In lower side of left wing, there is reflector which reflects IR on playground. Reflector is made of aluminum cooking foil.



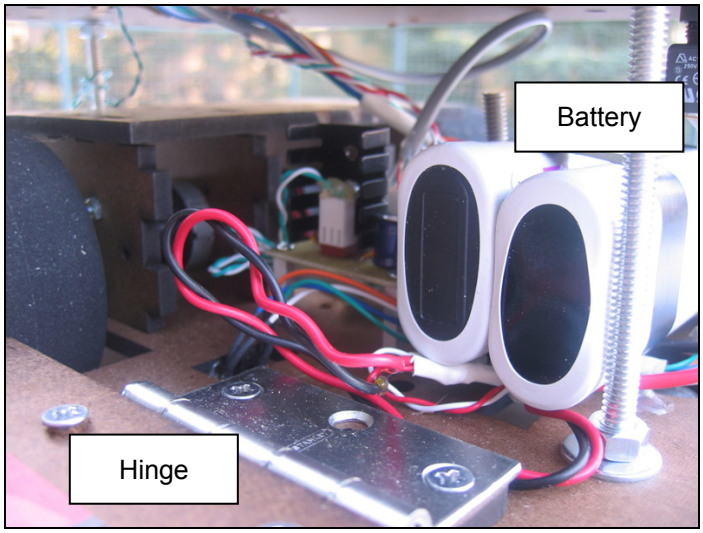
C. Circuit Supporter

Circuit supporter supports logic board and C32 board. Circuit supporter is attached to main panel with 3 bolt (1/4“ diameter). To adjust bolt location, there are several holes in circuit supporter and main panel.



D. Hinge

4 hinges are used to deploy wings.



E. Hinge Supporter

Because hinges are connected to wing and main panel by screw of ½ " long, we need hinge support to have enough space for screw. In every connection, 1 hinge supporter is used.

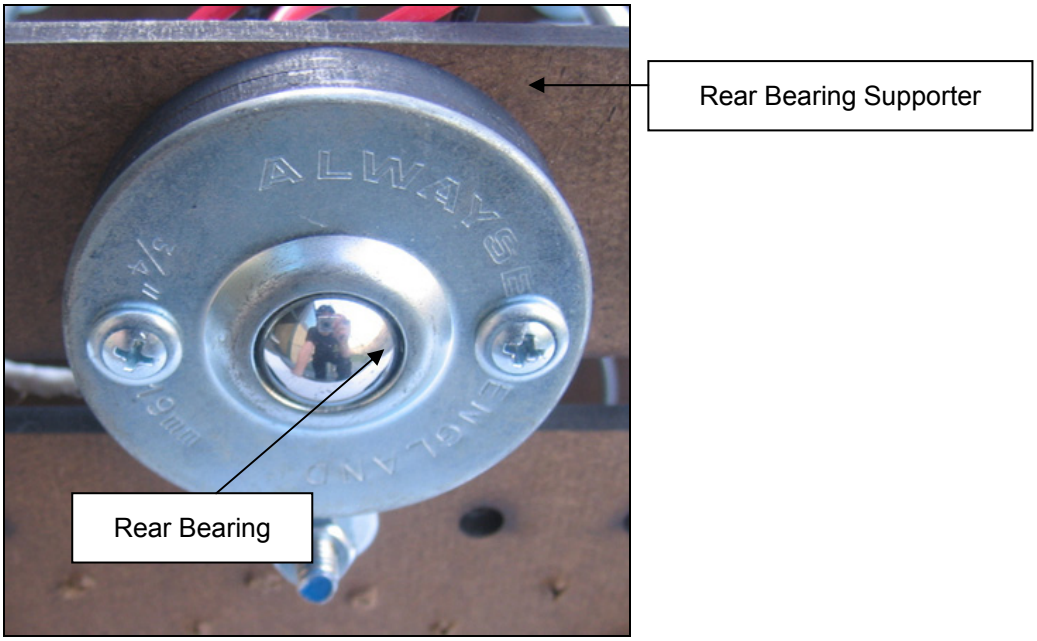
F. Front Wheel

Front wheels are chosen to have maximum grip force. At last lab (lab 8), we have some problem in friction of tire which make slow movement of robot although motor power is

enough. Front wheel is connected to motor directly. Connecter was machined on the lather in PRL.

G. Rear Support – Ball Bearing

Ball Bearing is chosen for rear wheel. Ball bearing is located center of rear part of robot. We decide to use one bearing instead of two to reduce weight. Ball bearing has much weight in spite of its small size.

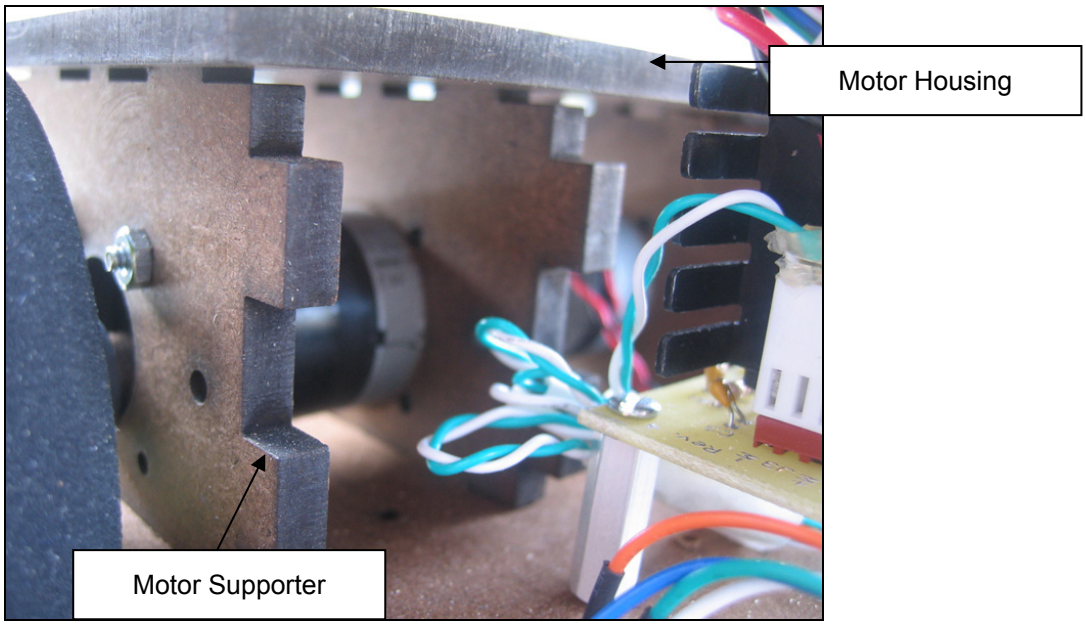


H. Rear Bearing Supporter

To level main panel, supporter is inserted between main panel and rear bearing.

I. Motor Supporter

Each motor is supported by 2 motor supporters. Motor supporter is connected to main panel and motor housing.



J. Motor Housing

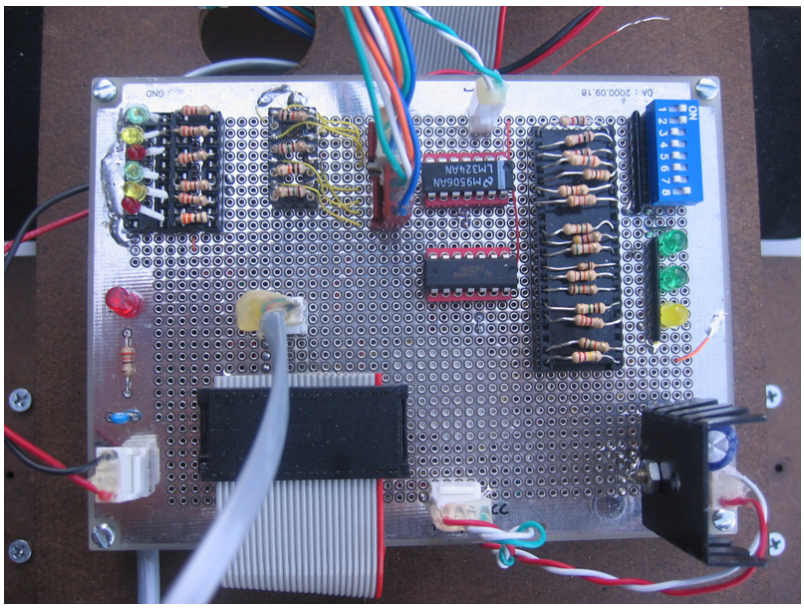
(2) Electronics

A. C32 Board

C32 board act stand alone process of robot. It located on circuit supporter.

B. Logic Board

All sensor signals are connected logic board. It performs signal conditioning and transmit signal to C32 board. Logic board also locates on circuit supporter. Logic board have voltage regulator to generate 5V for all circuit.

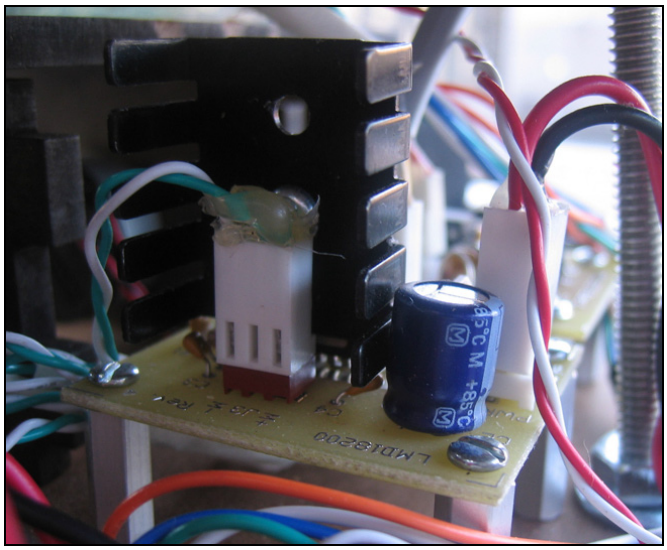


C. Tape Sensor

2 Tape sensors are located main panel. It located left side of panel and right side of panel to detect where is the robot is. Tape sensors are connected to logic board.

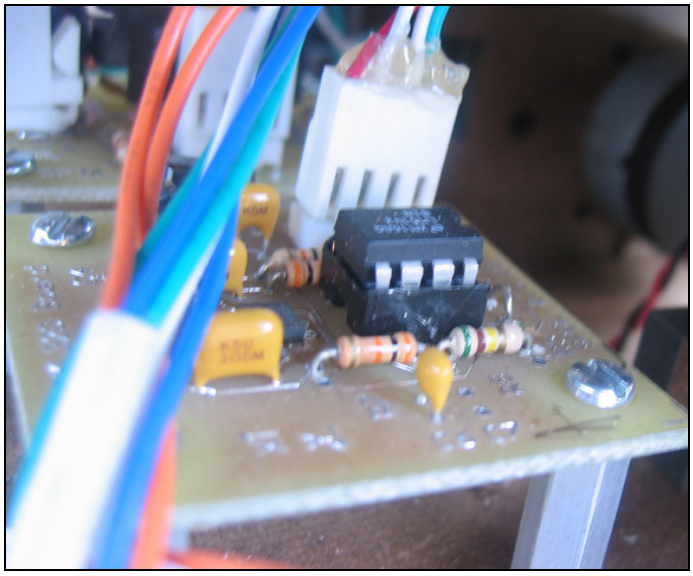
D. Motor Driver

Motor drive is connected to motor and logic board. To reduce distance to motor, motor driver is located on main panel, beside motor. Because 2 motor always move in same direction, just one motor driver is used. Motor driver get power from battery directly.



E. Accelerometer

To measure precise angle, accelerometer is located on main panel. It send angle of robot to logic board.



F. Motor

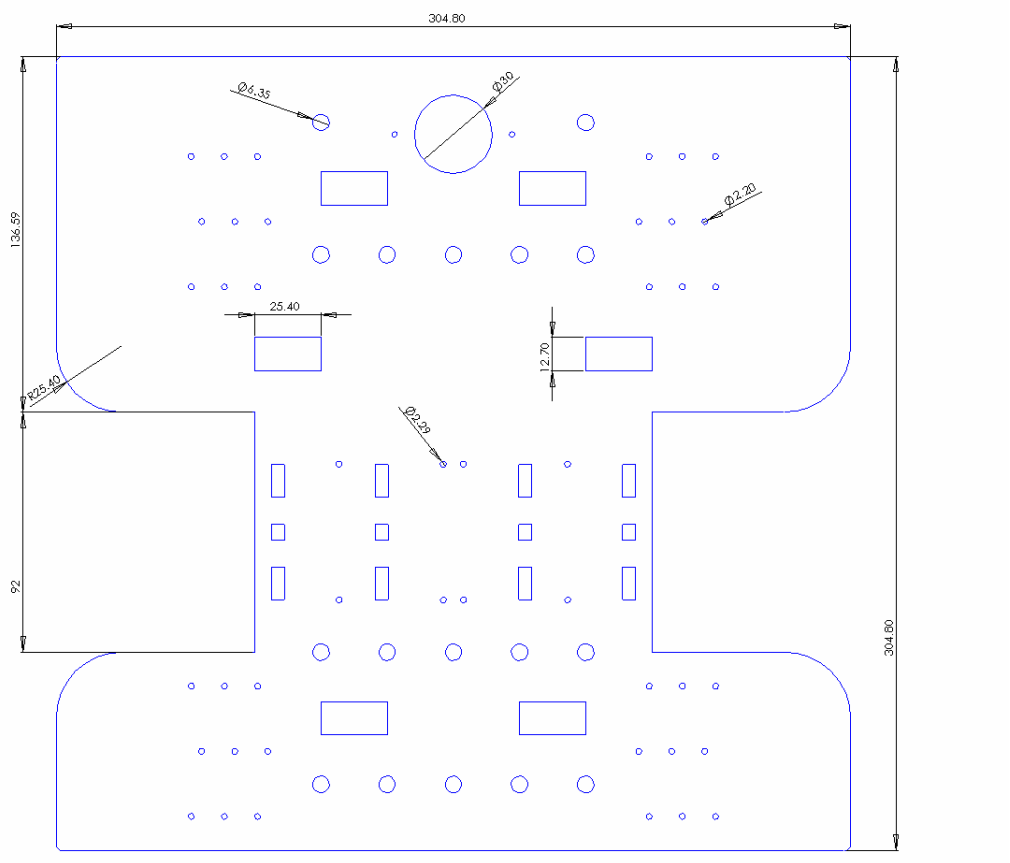
Motor drive wheel and connected wheel directly.

G. Battery

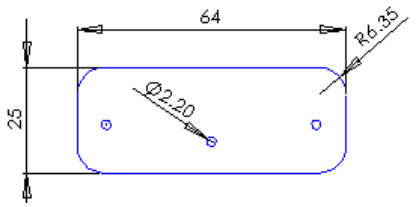
6-cell 7.2volt batteries are used for power robot. 2 batteries are connected in serial and make 14.4V. All logic circuit used 5V generated in voltage regulator in logic board except motor drive circuit which use 14.4V directly.

3. Drawings

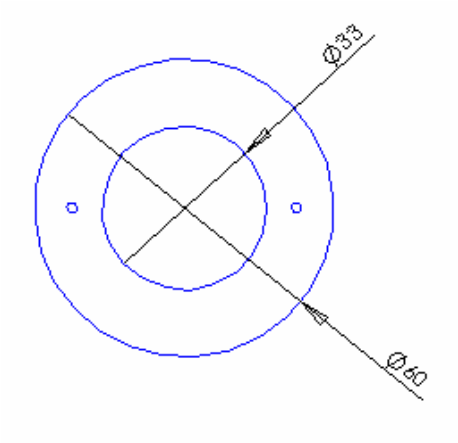
Main Panel 1EA



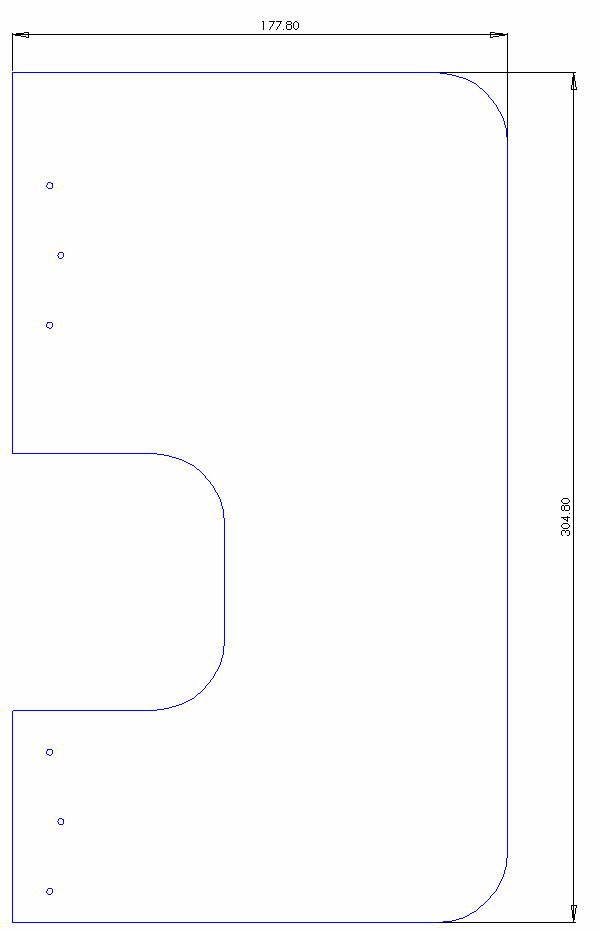
Hinge Supporter 8EA



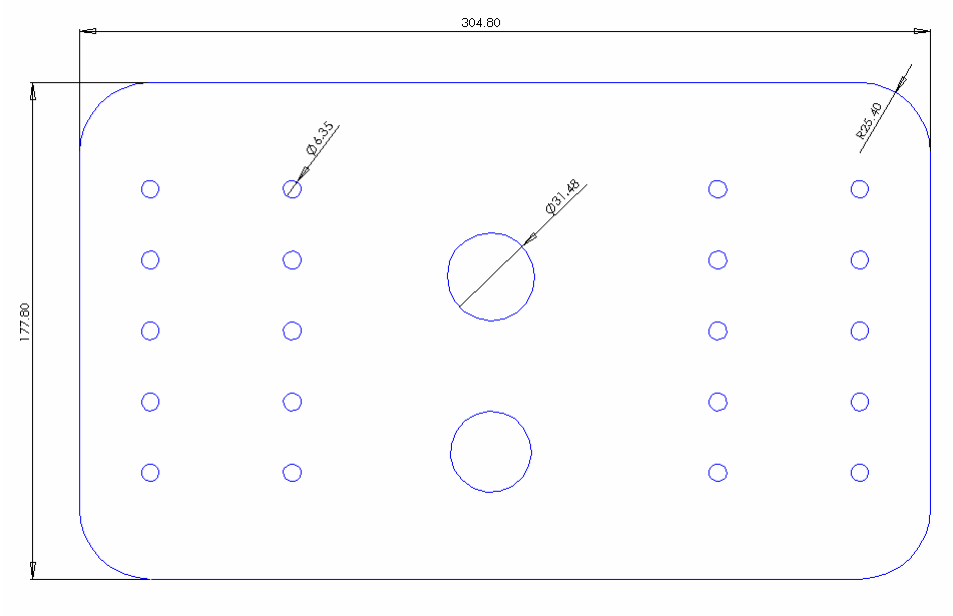
Rear Bearing Supporter 2EA



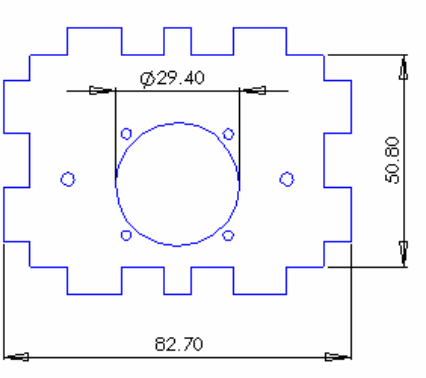
Wing 2EA



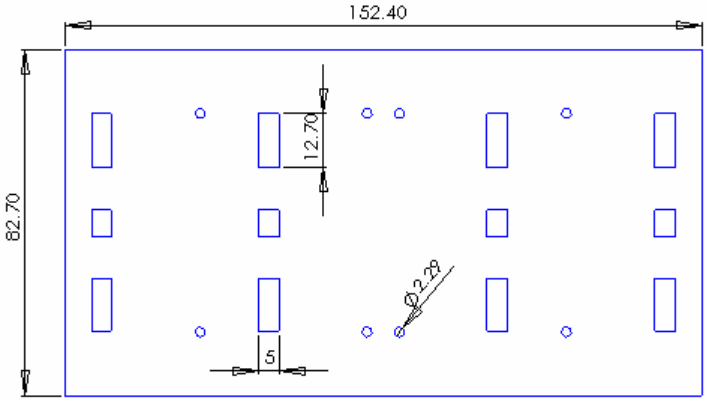
Circuit Supporter 1EA



Motor Supporter 4EA



Motor Housing 1EA

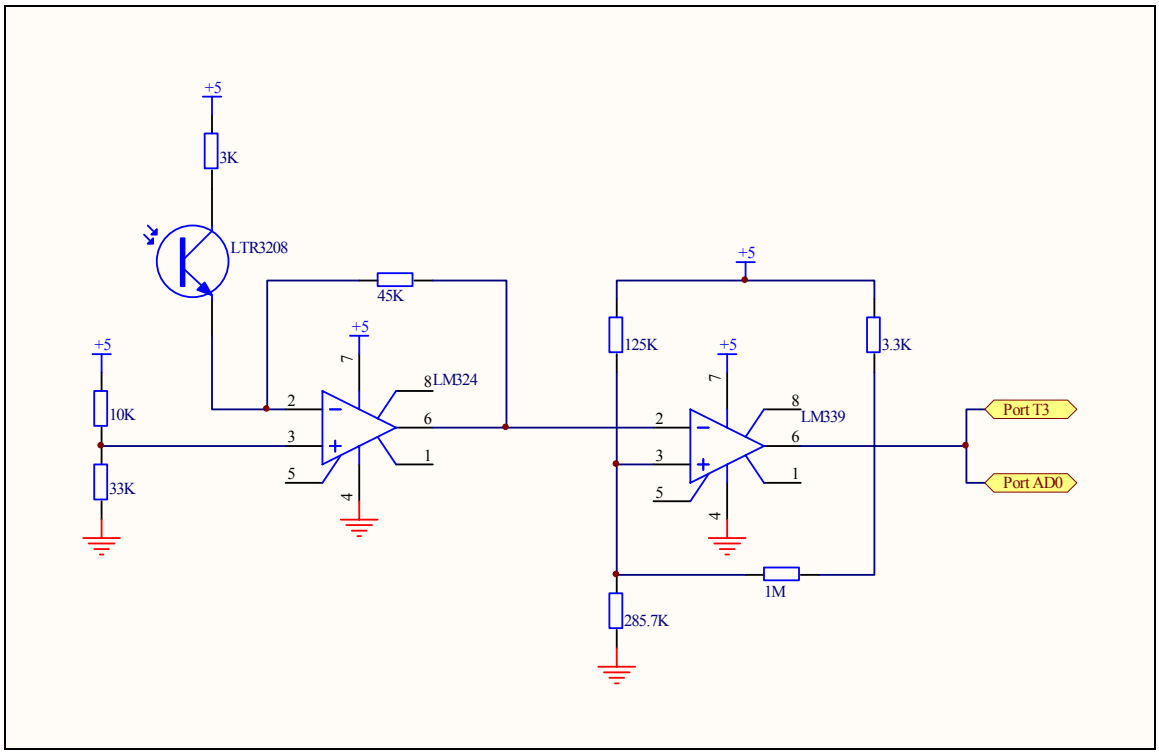


III. Circuit Design

1. Beacon Sensors

In order for our robot to be able to catch the information from the beacon in the field, it must be able to detect the signal coming from two beacons. Our team use only one beacon in the game, and we use the same sensor to detect the frequency, duty cycle and flash for the start. We have an accelerometer to measure the angle of the field, so we do not need to detect the beacon signal in the half of the field. Our approach is based on using only one IR sensor at the rear of our robot for detecting the flash and beacon behind our robot.

The schematic of the beacon sensor is given below.



We use the similar circuit of the lab2 of the ME218A. To be able to detect the signal from the long distance, we use one op amp(LM324). Since we use the inverting op amp, the output voltage is 3.8V when the input is LOW, but the output voltage is 3.0V when the input is HIGH. Also, we use a comparator(LM339) to digitalize the signal. The calculation of resistors are given below.

Let $V_{A1} = 3.6V$, $V_{A2} = 3.2V$, $R_{pull-up} = 3.3K\Omega$ and $R_3 = 1M\Omega$

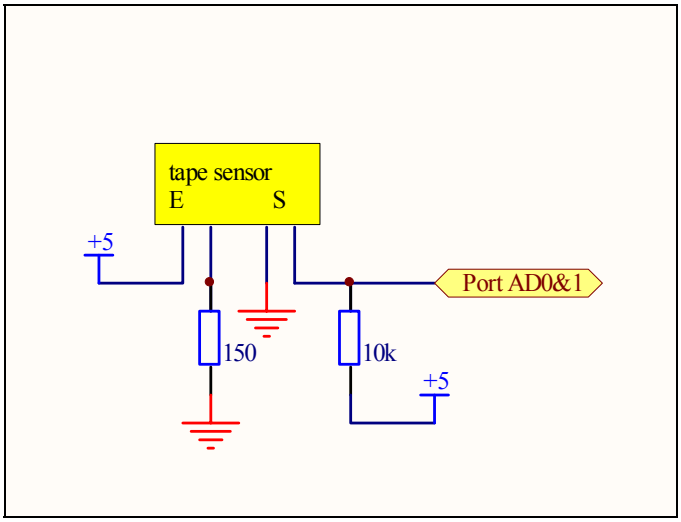
$$n = \frac{\Delta V_A}{V_{A2}} = \frac{0.4V}{3.2V} = \frac{1}{8} \text{ then } R_1 = nR_3 = \frac{1}{8} \times 1M\Omega = 125K\Omega$$

$$R_2 = \frac{R_1 \| R_3}{\frac{V_{CC}}{V_{A1}} - 1} = \frac{125K \| 1M}{\frac{5}{3.6} - 1} = 285.7K\Omega$$

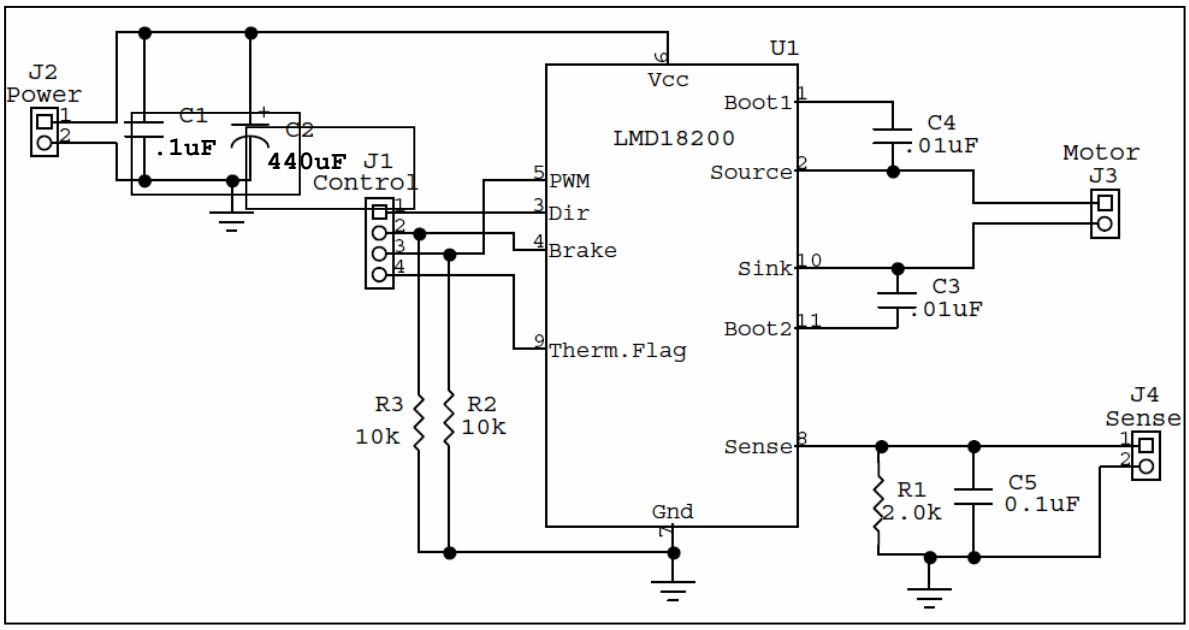
To deal with the signal from the sensor, we use two ports that are one T port and one AD port. The T port is for computing the duty cycle and frequency, and AD port is for determining the state of the signal, i.e., which is HIGH and LOW, and for detecting the flash for the start.

2. Tape Sensors

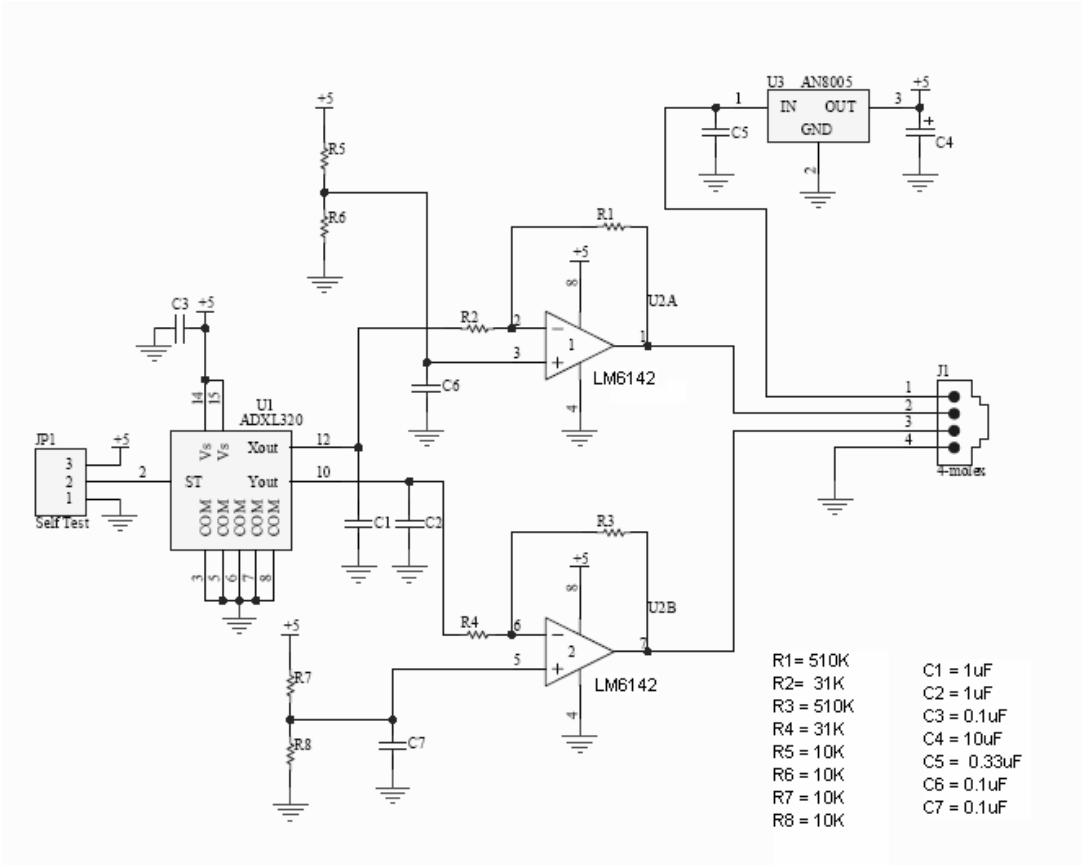
To detect the black tape in the field, we set up two tape sensors. In the previous lab, we should distinguish the black, white and the other colors. We only distinguish, however, black and the others in the current project. The schematic of the tape sensors is given below.



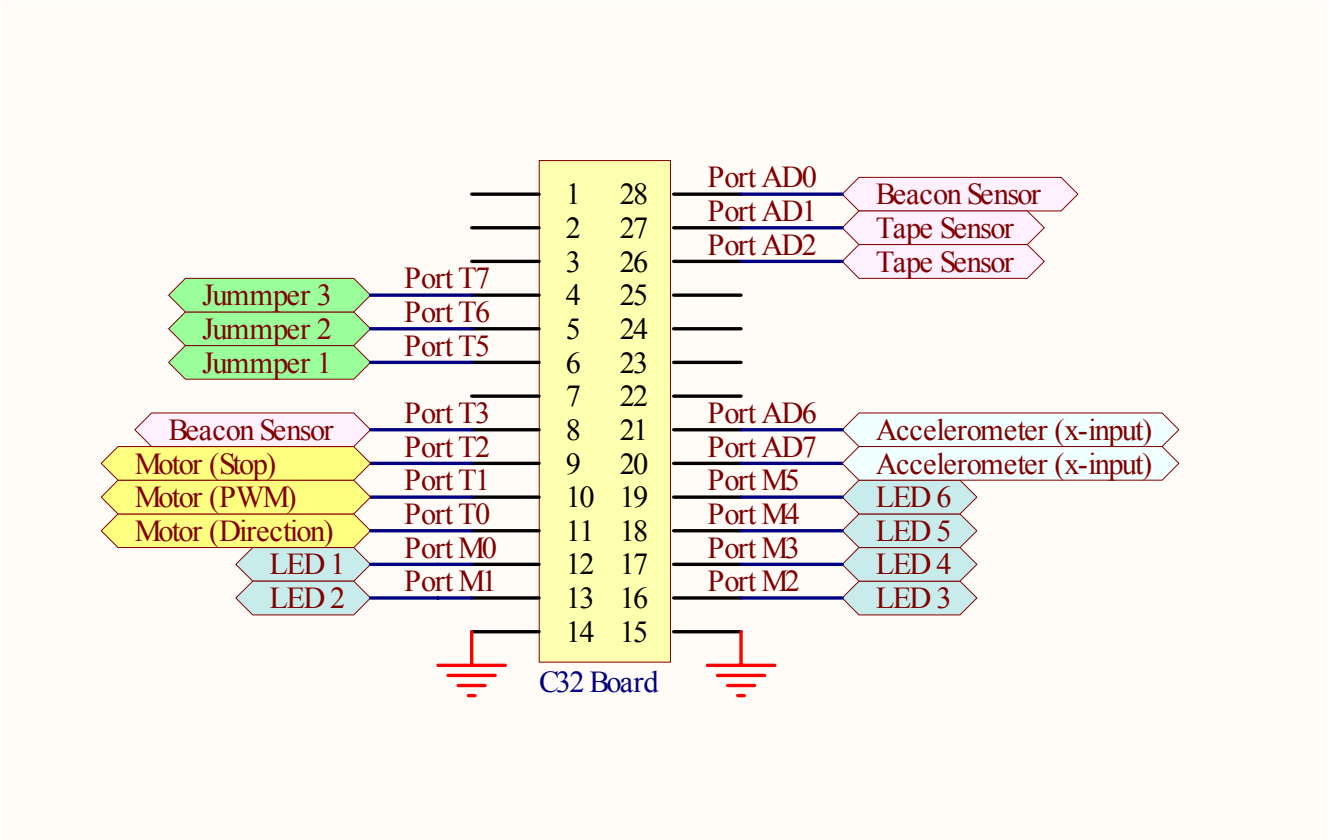
3. Motor drive



Accelerometer



4. Port assignment



IV. Software Design

1. Game Strategy

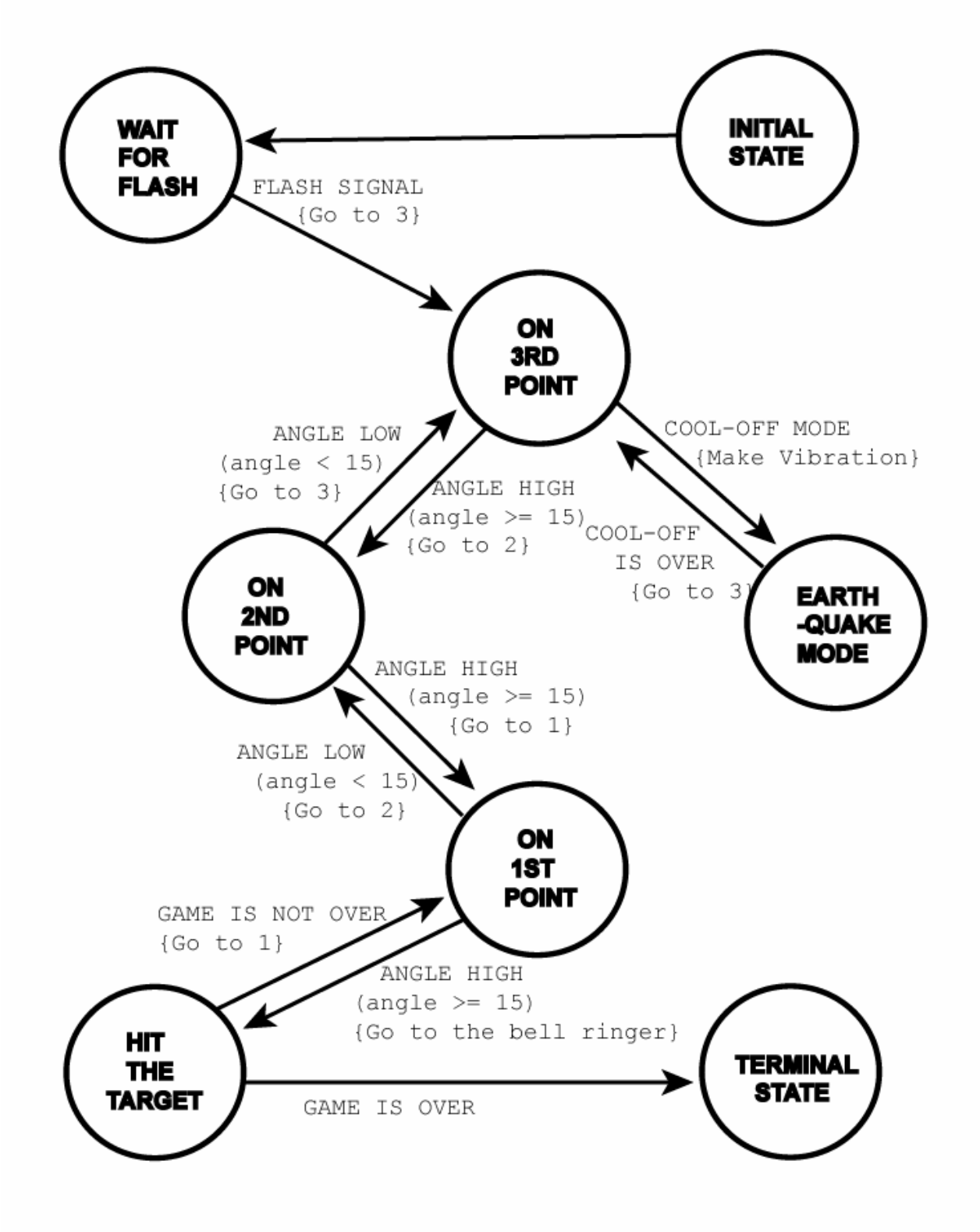
The most important information to determine if we move forward or backward is the value of angel which can be measured by the accelerometer. Fortunately the angle information of the accelerometer is fairy accurate enough to use. Thus, if the value of angle is equal or larger than 15 degrees in positive direction, we can believe that we have enough angles favorable to us and the bell ringer is on the right position.

Once the flash signal is detected and a game is begun, most of the movement of the robot is determined by the fact if we have enough angles to win the game. If we have the angle, the robot moves forward to the next point. And if not, the robot moves backward to the previous point to have a chance to move the central mass faster. On each point where a sensor is located to move the central mass, the robot takes a break at least for 0.5 seconds until we have enough angles to win. For these breaks the robot measures the angle of inclination because the accelerometer uses the direction of the gravity, therefore, it can output proper angle information only if there is no movement. And the break time is also useful in another way. Because the robot has a rest only on the sensing points, the central mass will take more time to move in direction favorable to us during the every 0.5 seconds of break time.

And we found that during the cool off (15 seconds break of the central mass) it is no use for the robot to stay in a same position. Rather than, it is better to move forward and backward to shake the field. If the opposite robot doesn't have a reliable system against this kind of confusion, we can possibly have luck to make the opposite robot fail to follow its intention in the confused situation. For this purpose, we implemented an earthquake mode which is activated during the cool off when the both robots are at position 3.

We also find that sometimes one attempt to press the button is not enough and fails. To supplement this weakness, we make the robot resume the ordinary routine after one attempt to press the button. With this strategy the robot will try the same attempts until it surely win the game.

2. State machine



3. Implementation of Software

(1) Basic module implementation

A. Beacon detection module

struct BeaconInfo BeaconDetect(void)	Detect the beacon signal and return BeaconInfo which has information of frequency and duty of the beacon signal.
int DutyRatio(int duty)	Normalize the value of duty with the raw data returned by BeaconDetect().
void DutyDisplay(int duty)	Display the duty on debug LED.

B. Line reader module

int DetectColor(int port, int thresh)	Read the voltage value of a specific port and judge which color the sensor is on by the value of thresh.
int IsBlackLineFound(void)	With the result of DetectColor(), it judges if the robot is really on the black line. To avoid false caused by noises or black spots on the field, it returns true only if the both sensors' signals are black at the same moment.

C. Accelerometer module

int GetAngle(int Vx)	With the raw data returned by GetAccData(), it calculates the estimated angle of inclination. The accuracy is 3 degree around the horizontal angle.
struct Point GetAccData(void)	Read the raw information of the accelerometer in x and y direction. To remove the effect of noise, it measures the value repeatedly 100 times and return the average values.

D. Motor drive module

void GoForward(void)	Apply full voltage to motor drive in forward direction.
void GoBackward(void)	Apply full voltage to motor drive in backward direction.
void Stop(void)	Cut all signals to motor drive and PWM module.

(2) State machine implementation

char RobotStateMachine(unsigned char event)	Determine which action will be applied and what state it moves to when a specific event appears. This function always keeps track of the current state by use of local static variables.

(3) Action implementation

void aGo3rdPoint(void)	This function is for moving to 3rd point directly from the initial point at the middle of the field. This action is called only one time at the beginning of a game. To find the 3rd point, it moves the robot until it founds the black line twice. After stopping, it move forward again till it found the black line again because it might miss the right position on the black line when it moves in high speed.
void aGoPreviousPoint(void)	Move to the previous point by moving forward. Sometimes the robot slides down and locates not on the black line, it passes through one black line if the angle is positive and the line detection is white at the beginning of the movement.
void aGoNextPoint(void)	Move to the next point by moving backward. Sometimes the robot slides down and locates not on the black line, it passes through one black line if the angle is negative and the line detection is white at the beginning of the movement.
void aEarthquack(void)	This function is implemented to shake and stir confusion during the cool off time. For 15 seconds, the robot move forward and backward twice

	between position 3 and 2.

(4) Supportive utilities module

void Delay(word time)	Delay the current state for the assigned time without. The unit of time is milli-second.
void DisplayDebugCode(int code)	Display a debug code on the LEDs which are used for debugging purpose.

(5) Initialization module

void AssignPorts(void)	Assign each port of C32 board for proper purpose and setting of hardware.
void InitTimer(void)	To use time related functions, set up the micro-processor initially.
void detectInitSetting(void)	To determine which side of the field the robot plays on and to use several different plans of strategy, jumper can be set before operation. This function detects the current setting of the jumper.

V. Expenses

Part	EA	Unit Price	Price	Vendor
Small Bolt	2	$0.85	$1.7	The Home Depot
Small Screw	1	$0.85	$0.85	The Home Depot
Tempered Hardboard 24x48	2	$6.59	$13.18	Minion's Lumber and Supply
Tail Wheel 1"	4	$1.66	$6.64	Hobby Shop
Tire	1	$22.99	$22.99	Hobby Shop
Metal Wire	1	$0.45	$0.45	Hobby Shop
Door Hinge	4	$3.29	$13.16	
Battary (11, 2EA, -5$)	1	$17	$17	SPDL
Motor Drive	1	$7.5	$7.5	SPDL
Telephone Connector	1	$1	$1	SPDL
Battary (11, 2EA, -5$)	1	$5	$5	SPDL

total **$89.47**

VI. Gems of Wisdom

- **Make the software into many modules.** It is much easier to test a single action in a modular test harness than several actions simultaneously. Also, it is helpful to test each module efficiently.

- **Make the wire connection short and neat as possible as you can.** It will reduce noise and you can find the error easily. The error due to the hardware, especially the wire connection, is quite popular but hard to debug, so if you are able to make them clearer and easier, then you can reduce the unnecessary time to find the trivial mistake.

- **Make the switch that can stop the whole system.** Even though the circuits work as expected in the test bench, no one can guarantee that they work the same on a moving robot. In that case, the switch, we call it “emergency stop”, will keep all of your circuits and robot safe.

- **Do not spend too much time to make some analogue circuit**, such as a noise reduction circuit. You can do the same thing through the software, i.e. the digital logic. The digital method is more powerful than the analogue one in many cases because it is easier and more reliable, thus it can save your time.

- **Conserve the circuit or code that works well during the process.** One day it works well, but after some modifications it does not. If you did not keep your well-working stuff, you should spend much time to make the same thing that works before the modifications. Also, after you have a platform that mostly works, don’t make more than one change at a time. If your platform does not work as expected with several changes, you can not find what change makes the error. The change to the platform must be deployed one by one.

- **Do not overcharge the battery.** On the presentation day, we wanted to full charge the battery, but the battery was broken due to the overcharge. Read carefully the instruction written on the battery charger and follow them strictly. Or you will buy the battery again and again.

- **Build robust hardware as early as you can.** Without robust hardware, the circuit and the software can not produce their full performance. And if the hardware is robust enough, it will cover some weakness of the circuit and software.

- **Have a good schedule plan.** With the time line the efficiency will be maximized. Without it, you will be tied down by the late business. It’s a good idea that one of members has the duty of managing the time schedule.

- **Less is More and Simple is the best**. Simplicity can make it possible that developers easily understand what the whole system is doing, what happen to the system. Also in a certain time limit, simplicity also lets developers implement a reliable system. Too much complexity demands too many challenges.