RC Car Competition

Designed by Students, For Students

Competition Sign Up Deadline: Continuous…. Come Join!

Competition Showcase Date: Saturday March 21st 2014 (Tentative)

The Waterloo Aerial Robotics Group – RC Car Competition

Background

The goal of the RC Car competition is to motivate potential recruits to participate in the normal activities of the Waterloo Aerial Robotics Group (WARG). The team leads of WARG have come to realize that it can be difficult for recruits to immediately jump into the day-to-day activities of WARG without having any prior knowledge or experience with the technology. It's a very common occurrence for new recruits to come to the team with very little to no knowledge of the skills needed to immediately contribute, but WARG has a fundamental goal of teaching any student the skills, if they wish to learn.

This competition has been developed by WARG as a tool to teach students about the following:

- Basic Programming Skills
- Control Theory
- Problem Solving
- Gain Tuning (Stabilizing a Controlled System)
- Path Following and Path Planning

At the end of the competition, it is WARG's goal that each participating student has learned how to perform a majority of the topics listed above. The competition has been designed to guide the recruits through the process, and for this reason, each phase of the competition relies on the previous phase to a certain extent. In other words, as the recruit proceeds through each phase of the competition, they are in fact building further upon their control program. Once the last phase is reached, the recruit should have a fully functional control program. How well this program functions will be based on the recruits time, effort, and general understanding of the skills WARG is trying to teach over the course of the competition.

Furthermore, the skills that students develop while participating in this competition can be immediately transferred to the normal activities of WARG. As well, these newly acquired skills can be advertised by the students during interviews, where they can inform the employer on what they've accomplished while participating in the competition.

The end goal of the competition is to develop a program that will make the RC Car autonomously follow a predetermined path. This is in fact exactly what is done by our airplane (SPIKE) in order to complete its competition. Our aircraft can always be improved upon, and it is our hope that students become interested in assisting the team with the skills they acquire through this competition.

Registration Fee

*** There will be a $5 fee in order to sign up for the competition, but the amount of money gathered by WARG will simply be put back to the participating students registered in the competition. To be an "official" member of WARG, the constitution of the team states that each member must pay an initial $5 fee. In order to incorporate this fee, as well as make the competition day a little more enticing, WARG is using this collection for the sole purpose of buying items such as pizza, pop, snacks, etc. for the final Competition Showcase Day.

The Competition

In teams of 3 or 4 (ideally), students will compete to control a remote control (RC) car that has been outfitted by WARG to have autonomous capabilities. Each team will be competing against the others for a prize pool (prizes to be determined and announced based on participation), where the top 3 teams will be rewarded respectively.

* Note that individual sign up is more than welcome, and teams of individuals will be generated

The competition will be graded as follows:

Phase	Task	Skills Acquired	Weighting
I	-Drive Forward 10sec -Drive Backward 10sec -Turn Right 10sec -Turn Left 10sec	Can you *call* a function? Can you make a loop?	2.5% each (10% total)
II	-Drive Forward 10m -Drive Backward 10m	Can you *make* a function?	5% each (10% total)
III	-Drive Forward 5m, then Turn Right 90 degrees then Turn Left 90 degrees	Can you think in terms of Heading?	15%
IV	-Drive a Straight Line while Maintaining Heading Stop after _(100)_m	Can you put everything together and create a Control Program?	15%
V	-Drive from Point A to Point B Stop within a _(5)_m Radius		15%
VI	Follow _(10)_ Waypoints		35%

Notes:

- Basic programming concepts will be explained to new recruits in order to help them succeed
- Values given in the _(Value)_ form are tentative and subject to change
- Phases 1 and 3: marks are awarded based on performance of the described task, not preciseness to the defined values (i.e. if you turn further than 90deg we will not punish you, but you must turn at least 90deg)
- Phases 2, 4, 5, 6, 7, 8: will be marked based on your team's error, relative to the other teams
- The system has been set up to relay GPS information back to WARG, we will use this information to determine your error in accordance to the given task. This eliminates discrepancy with regards to GPS drift (i.e. if we were to mark a GPS coordinate on the ground and say "go here," the GPS may think that spot is in two different locations based on the number of satellites it has at that given time)

Terminology:

- Heading: a 360 degree value of the direction to North (basically like a compass with N = 0deg and 360deg, E = 90deg, S = 180deg, and W = 270deg)
- Waypoint: a GPS coordinate described by a Latitude and a Longitude

What You Need To Know

There are a few things you'll need to know and understand (other than basic programming concepts) in order to control the RC Car.

What to Download

WARG has created presentational step-by-step presentations to help you setup the programs needed:

MPLAB X: (required)

https://docs.google.com/presentation/d/1XvRea8qPeCgIOqU_4q_P-z-q1EOupfJyO801Is2gDz4/edit?usp=sharing

This is the IDE used to program the chip on the RC Car

Github: (optional)

https://docs.google.com/presentation/d/1HkjCdRQyY1LFaUO-24SxHGLx4cgeq_Z5OWQQNKFfidM/edit?usp=sharing

*Please note that the Github presenation presentation might be out of date, but the same concepts will apply in order to get it setup on your system. Contrary, you might not need to download the desktop client, and can simple get the RC-Car-Bootstrap-Competition project as a .ZIP, in which case you can place it where you want and open the Project from MPLAB.

*If you choose to use Github, you DO NOT need a UWARG member to add you to the group. Since the project is Public, anyone can view and download the RC-Car-Bootstrap in order to start programming. See the next section for instructions on how to do so.

Where to Get the Code (GitHub)

The following link is where you can download the base project for the RC Car Competition:

https://github.com/UWARG/RC-Car-Bootstrap-Competition

On the right side of the screen, select "Download ZIP." Place the unzipped folder somewhere on your computer for you to access. Open this folder as a Project in MPLAB X, and you should be ready to start coding.

*Note: A presentation may be created later to teach people how to use MPLAB X, but if you start typing some code and want to test that it compiles (i.e. you've created correct code), then you may press the "Build & Clean" button to initially test the program. It should tell you if there are any errors with your code, or something that it doesn't understand.

Functionality Developed by WARG for You to Use

WARG has developed a bunch of functions that perform hardware communication with the car. This means that the recruits do not need to understand how to truly talk to the devices, but instead can program the system at a much higher level. What this means is something similar to the following comparison. Somebody tells you to shout “Hello!” You don’t truly think about the fact that all those fancy biology components in your body are doing things, and somehow in the end…. magical noise comes from your mouth. What happened is that somebody gave you a command at a “higher level,” your brain then took that command and did the appropriate things without you truly knowing what’s going on, in order to make “Hello!” come from your mouth. Might be a weird analogy yes, but that’s similar to what’s going on in the code.

WARG has given you the following functions to work with. They must have the specified inputs, and they perform the specified actions:

setSteering(int percent);

This function does exactly what you think it does. It takes an int value input ranging from -100 to 100. With -100 turning the wheels fully to the left, and 100 turning the wheels fully to the right.

setThrottle(int percent);

This function does exactly what you think it does. It takes an int value input ranging from -100 to 100. With -100 being full speed in reverse, and 100 being full speed forward.

getTime();

This function returns the time in milliseconds as type *long int*, the function takes no inputs.

You are also given a struct to work with that holds and updates all of the appropriate GPS Data. In order to access this information, simply type “GPS.______” where the ____ is filled appropriately as follows:

GPS.latitude;   // the current latitude as type *long double*

GPS.longitude;   // the current latitude as type *long double*

GPS.speed;   // the current ground speed as type *float*

GPS.heading;   // the current heading as type *int*

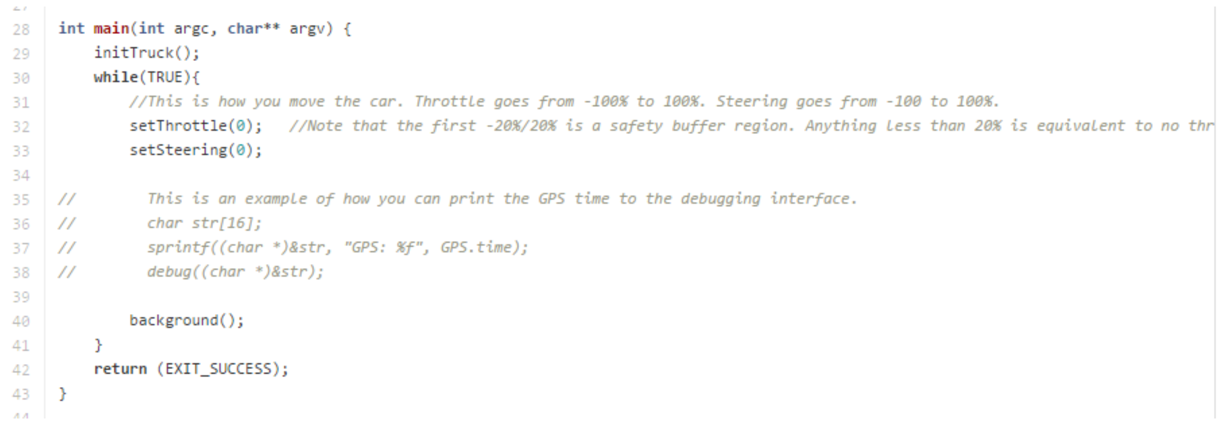
GPS.satellites;   // the current number of satellites as type *char*

GPS.positionFix; // if there is a position fix or not as type *char*

GPS.altitude;   // the current altitude as type *int*

Where You Program in the Code

When you download all of the required programs, and get the RC-Car-Bootstrap from Github, you will see the following code segment in the main.c file.



```c
int main(int argc, char** argv) {
    initTruck();
    while(TRUE){
        //This is how you move the car. Throttle goes from -100% to 100%. Steering goes from -100 to 100%.
        setThrottle(0);   //Note that the first -20%/20% is a safety buffer region. Anything less than 20% is equivalent to no thr
        setSteering(0);

//        This is an example of how you can print the GPS time to the debugging interface.
//        char str[16];
//        sprintf((char *)&str, "GPS: %f", GPS.time);
//        debug((char *)&str);

        background();
    }
    return (EXIT_SUCCESS);
}
```

You will notice that there is already some code in the main body…. YOU MUST LEAVE THIS THERE.

The code that exists is there to initialize the system and ensure it is running properly. As well, it performs all of our background tasks that send data do us about what your system is doing at the moment. All of your code must take place after “setSteering(0);” and before “background();.” Please note that this is meant by your code in the main function of your program, later on in the phases, you will be required to develop your own functions, which must be developed outside of this main.

Aspects of Programming (Super Simplified)

******WARNING: THIS SECTION IS NOT COMPLETED, AND YOU'LL NEED TO KNOW A FEW MORE THINGS THAN WHAT IS DESCRIBED HERE. FOLLOW THE PROVIDED TUTORIAL LINKS TO HELP YOU LEARN FOR THE TIME BEING.***

For any recruit who is nervous, unsure, or thinking twice about joining the competition due to not having prior (or having minimal) knowledge of programming, WARG would like to motivate you to get past that fear. SO GET MOTIVATED!!!

Programming may take a little bit in order for the concepts to finally "click," but once you grasp it…
…it all seems so simple

You will be programming in a language known as C. Just as there are multiple verbal languages (i.e. English, French, Spanish, etc.), there are a bunch of programming languages. These programming languages are a lot easier to transition between and pick up if you know one of the languages, in comparison to verbal languages.

Moving on, here is a REALLY REALLY simplified generalization of some programming concepts.

Variables

You can think of variables just like the ones you'd use in math, by this we mean that they can be seen as "place holders" for a value. Meaning that in math you may declare:

"Let x be the number of apples Brian has on Monday"

Variables in programming are similar, and are place holders for actual values. These values are hidden away and saved somewhere in the microchip (in memory…. but we don't need to know about this right now). Here are a couple examples of variable *types*:

Type	***Description***	***Example***
int	Can be thought of as a *whole* number, or a number *without* decimals	1, 16, 3054
float	Can be thought of as a number *with* decimals	1.02, 506.23579

You totally thought the list was going to be longer didn't you. Well, technically it is quite a bit larger yes, but in all honesty you could write the whole code for the competition using just these variables and you would be fine. With that said, you might want to utilize something known as an *Array*. An array can be thought of as a list of a certain type of variable. If you want to save data on a few things that may relate, or be of similar characteristic, to one another… you may want to put them into a list.

In order to initialize your variables in your program, you would do as such:

```
int x = 0;    //This means that you've declared a new variable "x", that is of type int, and has starting value 0
float myFloat = 2.34; //a new variable "myFloat", of type float, with starting value 2.34
int myArray[5];  //Declaration of an array (un-initialized)…
```

…This means that you've declared an array of 5 int variables (i.e. there is a list of 5 int's), you've called this array "myArray", and you did not initialize the values to anything

You would then later access individual members of the array by doing the following:

```
myArray[0] = 8; //This sets the first member in the list to hold a value of 8
```

Piece of cake right? No? Well that's ok, because it took some of the Lead members of WARG a little while to truly get a grasp as well.

*** Obviously there's more to programming then just declaring variables, and WARG is currently in the process of writing this document to better help recruits get involved! For the time being, please look at the following tutorials online regarding basic C programming. There's lots of things on the internet if you use that thing…. what are the kids calling it? "The Google?"

You'll need to know about "if" statements, "while" loops, and possible "for" loops in order to successfully complete the phases in the competition. Here are a few useful links to help you:

http://www.cprogramming.com/tutorial/c/lesson1.html -- Specifically:

- http://www.cprogramming.com/tutorial/c/lesson2.html // if statements
- http://www.cprogramming.com/tutorial/c/lesson3.html // looping
- http://www.cprogramming.com/tutorial/c/lesson4.html // functions

Or more importantly for now (considering we're still developing this document), in order to be a part of any student team, you need to be able to research. Therefore the following is the most important link we can give you: http://lmgtfy.com/?q=c+tutorial

Phase Definitions

This section of the document is meant to further define what is required, and expected for each Phase of the competition.

Phase I

Purpose:

Allow recruits to get acquainted with a very simple program, and how the process works to program the RC Car. The recruits will need to call functions that have been premade for them, and make a simple loop.

Required Tasks:

- Drive Forward for 10 seconds
- Drive Backward for 10 seconds
- Turn Right for 10 seconds
- Turn Left for 10 seconds

Explanation:

This phase should not require further explanation in order to get a grasp on what's required. By defining a few variables, and using the pre-made functions given to use, the recruit should be able to create a program that performs the above tasks.

Evaluation and Weighting: 10% in total for the phase

2.5% will be given per each of the 4 tasks

It is simply expected that the car moves in the appropriate direction for the given time period. The car does not need to stop "spot on" the 10 second mark, but WARG will be able to easily tell if you've implemented the correct functionality in your code. It is also key to note that care should be taken into consideration, and there is no need to put the car at 100% throttle in order to complete this task.

Phase II

Purpose:

The initial purpose of this phase is to have the recruits create a new function from scratch. Once the recruits have completed later phases, and gained further knowledge, they can return to this phase in order to improve it and receive a higher weighting.

Required Tasks:

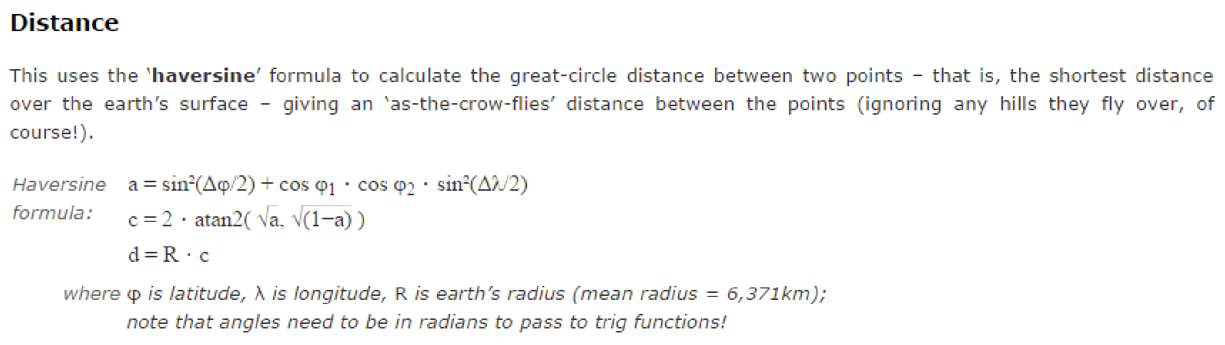
- Drive Forward 10 meters
- Drive Backward 10 meters

Explanation:

In this phase the recruit must create their own function in order to determine the distance travelled from the car's current position, to a start point (or previous waypoint).

Your function definition should pass in latitude and longitude values of the GPS coordinates (2 sets, one for point 'A' and one for point 'B'). The following website gives you a pretty good layout of how your code should look (to some extent, it will look different due to you programming in C and not Java Script). Specifically the following code bite as well:

http://www.movable-type.co.uk/scripts/latlong.html



Distance

This uses the '**haversine**' formula to calculate the great-circle distance between two points – that is, the shortest distance over the earth's surface – giving an 'as-the-crow-flies' distance between the points (ignoring any hills they fly over, of course!).

Haversine formula:

$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$

$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$

$d = R \cdot c$

where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km);
note that angles need to be in radians to pass to trig functions!

Above is the formula you must program into your function.
Below is how the created such a function with Java Script. *** Your function should return a float of "d"

```
var R = 6371; // km
var φ1 = lat1.toRadians();
var φ2 = lat2.toRadians();
var Δφ = (lat2-lat1).toRadians();
var Δλ = (lon2-lon1).toRadians();

var a = Math.sin(Δφ/2) * Math.sin(Δφ/2) +
        Math.cos(φ1) * Math.cos(φ2) *
        Math.sin(Δλ/2) * Math.sin(Δλ/2);
var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

var d = R * c;
```

The only addition you must make to this code (besides writing it correctly for your C program), is to multiple the "c" value above by ±1000. Is needed to convert kilometers into meters.

Evaluation and Weighting: 10% in total for the phase

5% will be given per each of the 2 tasks

Initially, it is simply expected that the car moves in the appropriate direction for the given distance. The car does not need to stop “spot on” the 10 meter mark, but WARG will be able to easily tell if you’ve implemented the correct functionality in your code. It is also key to note that care should be taken into consideration, and there is no need to put the car at 100% throttle in order to complete this task.

Furthermore, out of the 5% given towards either task, a weighting of 4% will be achieved by implementing the functionality described in the previous paragraph. The additional 1% will be achieved based on error to the 10m mark. In order to reach the 10m as close as possible, the recruit must implement proper Control Theory in order to slow the car down and receive the extra weighting. The code cannot be written to simply reverse the car for a millisecond after reaching the 10m mark, as this harms the equipment and decreases its lifespan. You may not reverse back to the 10m mark either if you have passed it. Additionally, in order to achieve this extra weighting (and so that WARG can see your control loop is working correctly), the RC Car must be started at an initial throttle of _(40)_%. The weighting will be received based on error in comparison to the top team for this phase (team closest to the 10m mark).

Please note that the control theory aspect will be learned in later phases of the competition. The “closeness” to the 10m mark is worth only 1% since it is WARG’s desire that all teams initially develop the code well enough to accomplish the initial 4% of this phase for each task. If teams finish the majority of the competition, and after learning control theory, they may return back to this phase in order to try and receive the addition 1%.

Phase III

Purpose:

The initial purpose of this phase is to have the recruits begin to think in terms of heading, and how it relates to controlling the steering of the car. Once the recruits have completed later phases, and gained further knowledge, they can return to this phase in order to improve it and receive a higher weighting.

Required Tasks:

- Drive Forward 5 meters, then
    - Turn Right 90 degrees, then
    - Turn Left 90 degrees

Explanation:

In this phase the recruit must create their own functionality in order to determine which direction to steer the car at any given point in time. The car may bounce around, and therefore move off of the desired path. Since we are constantly checking the state of the car (by continuously looping through the entire code), we can determine which way to steer the car at any point in time. It is recommended that this functionality be developed in the form of a function.

At no point during this phase should the car come to a stop while transitioning between task objectives. The desire is that the car drives a smooth path. For this reason, the car should never operate under _(20)_% throttle while transitioning between task objectives. The car may only be controlled to a complete stop once you are completed the phase in the "Turn Left 90 degrees" sub-task. Again, the car may never be driven backwards, and must always be moving forwards.

Evaluation and Weighting: 15% in total for the phase

12% will be given for initially achieving the tasks above by going (at minimum) the distance or angle change that's stated.

Initially it is WARG's desire that the recruit moves on after achieving this initial functionality, and returns back at a later time to improve upon the phase's functionality. The additional 3% will be achieved similarly to the methods described in Phase II, in the same manner that the additional 1% is achieved.

Phase IV

Purpose:

The purpose of this phase is for the recruit to put all of the functionality they've developed together, and create a simple control loop.

Required Tasks:

- Drive a straight line
    - While maintaining heading
    - Stop after _(100)_m

Explanation:

In this phase the recruit must create a control loop that will steer the car accordingly and keep it on the desired path. This involves monitoring the heading of the car, and steering the car in the appropriate direction if it starts to stray from the desired path. A heading a starting direction will be determined by WARG based on the competition testing location (the heading to hold will be hardcoded into the recruits program ahead of time, since at this point we are not expecting anything that calculates which heading should be held).

You will notice that if you turn the car too much in a given direction when it strays off the path, it will quickly be weaving back and forth on the path, and the amplitude of this "weave" will increase overtime. This is because you must *converge* towards the desired path, rather than do a jerking turn that makes the car immediately be on the path. The rate at which you turn the steering, relative to your distance from the path, is controlled through the *control loop*, and by tuning the gains appropriately to fit your system (in this case the system is the RC Car).

Please refer to the *Control Theory* section in order to gain more knowledge on how to create a control loop. As well, WARG is considering bring in lecturers in order to host short lectures / tutorials on these concepts. This will be done based on the interest of the students.

Evaluation and Weighting: 15% in total for the phase

Considering that we do not expect the car to stray to too far from the desired path (there are no strong winds pushing the car off its path, such as that which occurs with the airplane while in the air), we do not expect the error from the path to be very large at all. Even if the recruit poorly tuned that control loops / gains, we believe the system shouldn't have a problem staying on the desired path.

With this in mind, the 10% is allocated towards a large part of this phase. The amount each recruit receives will be based on their error in comparison to the top team's error.

The additional 5% of this phase will be calculated based on the recruit's error in distance travelled, in comparison to that of the top team's error. (Hint: this is a controlled stop like that described in Phase II)

Once again, you must have a starting throttle of at least _(40)_%, and this should be your "cruising" throttle while following the path. You may start to decrease and control your stopping point within 10m of the desired travel distance of _(100)_m. Again, the car should always be moving forward.

Phase V

Purpose:

The purpose of this phase is for the recruit to develop dynamic functionality of their code (i.e. not hardcoded values). This phase also has the purpose of developing code that drives the car based on waypoints.

Required Tasks:

- Drive from Point A to Point B
    - Stop within a _(5)_m radius

Explanation:

The functionality as described in Phase IV should be implemented in this phase in order to better complete the task described.

You'll need to apply some logic into your program and probably create another function to perform basic geometry. In order to find the line (and heading) you must follow, you'll have to generate a xy based system from you latitude / longitude coordinates. You can do this by specifying an anchor point (of latitude longitude, this could possibly be your position on start-up). You can then find your x component, and y component of any given waypoint by holding either your latitude / longitude constant, and sending your anchor, and new waypoint through your getDistance() function.

On competition day, WARG will provide the recruit with a Waypoint to travel to. The program must get its current GPS position (current waypoint), and then calculate a straight line from this current position to the desired waypoint given by WARG. This is the heading that the car must be desired to hold in order to get to the waypoint.

WARG will start the car in the opposite (or some determined) direction relative to the goal Waypoint in order to ensure the system is calculating things appropriately.

*Hint: The RC Car should calculated which direction to turn in order to be going the appropriate heading the fastest.

Evaluation and Weighting: 15% in total for the phase

10% is allocated towards a large part of this phase. The amount each recruit receives will be based on their error in comparison to the top team's error. Error is determined by the cars distance from the desired path throughout the duration of the phase.

The additional 5% of this phase will be calculated based on the recruit's error in distance from the _(5)_m radius circle of the waypoint (i.e. if the further you are from this radius, the more error you have), in comparison to that of the top team's error. If you stop within _(5)_m, you will be awarded with the full 5% weighting.

Once again, you must have a starting throttle of at least _(40)_%, and this should be your "cruising" throttle while following the path. You may start to decrease and control your stopping point within 10m of the desired waypoint. Again, the car should always be moving forward.

Phase VI

Purpose:

The purpose of this phase is to combine everything that the recruit has learned in order to make the RC Car follow a dynamic path of waypoints. Two paths of _(10)_ waypoints will be given for the car to follow (the car will follow one path, and then the new waypoints will be programmed, and the second path will be followed).

Required Tasks:

- Follow a set of _(10)_ Waypoints

Explanation:

At this point in time, WARG hopes that the recruit has learned enough from previous stages in order to accomplish the described task of following a set of _(10)_ waypoints.

Waypoint 1 will be the RC Car's current position, the additional 9 waypoints will be provided.

Evaluation and Weighting: 35% in total for the phase

34% is allocated towards a large part of this phase. The amount each recruit receives will be based on their error in comparison to the top team's error. Error is determined by the cars distance from the desired path throughout the duration of the phase.

The additional 1% of this phase will be calculated based on the recruit's error in distance from the _(5)_m radius circle of the final waypoint (i.e. if the further you are from this radius, the more error you have), in comparison to that of the top team's error. If you stop within _(5)_m, you will be awarded with the full 1% weighting.

Once again, you must have a starting throttle of at least _(40)_%, and this should be your "cruising" throttle while following the path. You may start to decrease and control your stopping point within 10m of the desired waypoint. Again, the car should always be moving forward.