

Robotic Car with Vision System
Design Review

Ruben Zhao, Brian Chung, Richard Otap
TA: Paul Rancuret

February 25, 2010

1 Objectives

A mobile robot will track and follow objects through the use of a camera mounted on a servo. More specifically, the goal of the project entails: (1) object tracking, (2) closed loop motion, and (3) real-time object following. Ultimately, the robot will be able to autonomously detect an object of interest and to follow the object. The project provides benefits to multiple spheres of interest, including home security, defense technologies, and the automotive industry:

- **Home Security:** Versions of this robot could identify intruders, track their movements, and in the future, disable them until law enforcement arrived.
- **Defense Sector:** The modern battlefield is steadily evolving to eschew the need for human soldiers. Instead, the presence of mobile robots has steadily increased. Our robotic platform could be modified to identify enemy combatants and track their movements.
- **Automotive Industry:** DARPA's first "Grand Challenge" was held in 2004, in which competing cars attempted to drive autonomously through a track. With the development of our platform, the automotive industry could advance to autonomous automobiles.

Features include:

- **Onboard computer:** The robot will have a computer onboard running Linux, allowing flexibility and speed in the types of programs that can be run.
- **Webcam running at 30 frames per second:** Able to capture and process video in real time, and fast enough to capture a variety of moving objects.
- **Pan and tilt:** The camera will be mounted on one servo motor, for panning the camera. This will allow it to keep the object in the center of the frame while the robot is moving.

2 Design

2.1 Block Diagram

2.1.1 Linux SBC

- The main form of control will be an single-board computer (SBC) running Linux. Due to the requirements, the computer will have UART for communication with the microcontroller as well as a USB port for interfacing with the web camera.
- The SBC will handle functions such as receiving data from the webcam, processing the data, and communicating with the motors via the microcontroller to produce locomotion and to orient the camera. The system will be closed loop in that the drive electronics will indirectly affect the cameras position, whereas the pan motor will directly affect the camera's position.

2.1.2 PIC microcontroller

- To enable the SBC to control the motors, one or more PIC microcontrollers will be used.
- The microcontrollers will act as an intermediary to translate serial data into control signals understood by the motor control circuits.

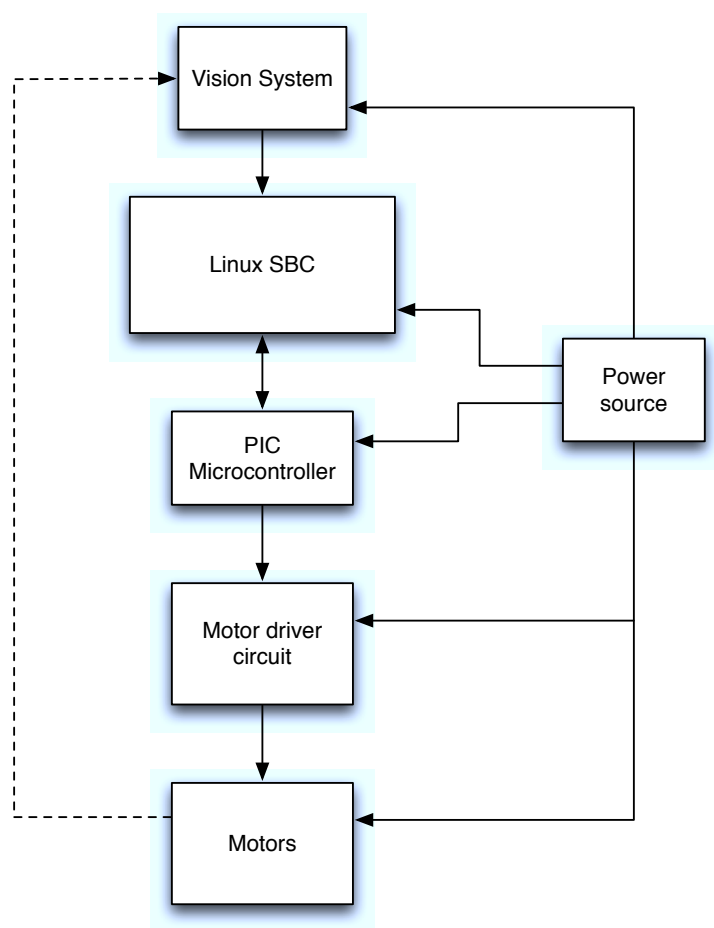


Figure 1: Block diagram of the system

2.1.3 Motors

- Movement of the robot will be accomplished via a dual-motor gearbox, controlled directly by the PIC microcontroller, and thus indirectly by the SBC.
- There is also a single servo motor, to center the camera on the target.

2.1.4 Power Source

- The power source will have to supply the control electronics as well as drive electronics.
- A power regulator will maintain a constant +5V for TTL logic and the SBC

2.1.5 Vision System

- The vision data will be provided through a web camera.
- The requirements for the camera will be relatively simple, requiring a minimal frame rate and resolution, and a USB connection.

2.2 Circuits

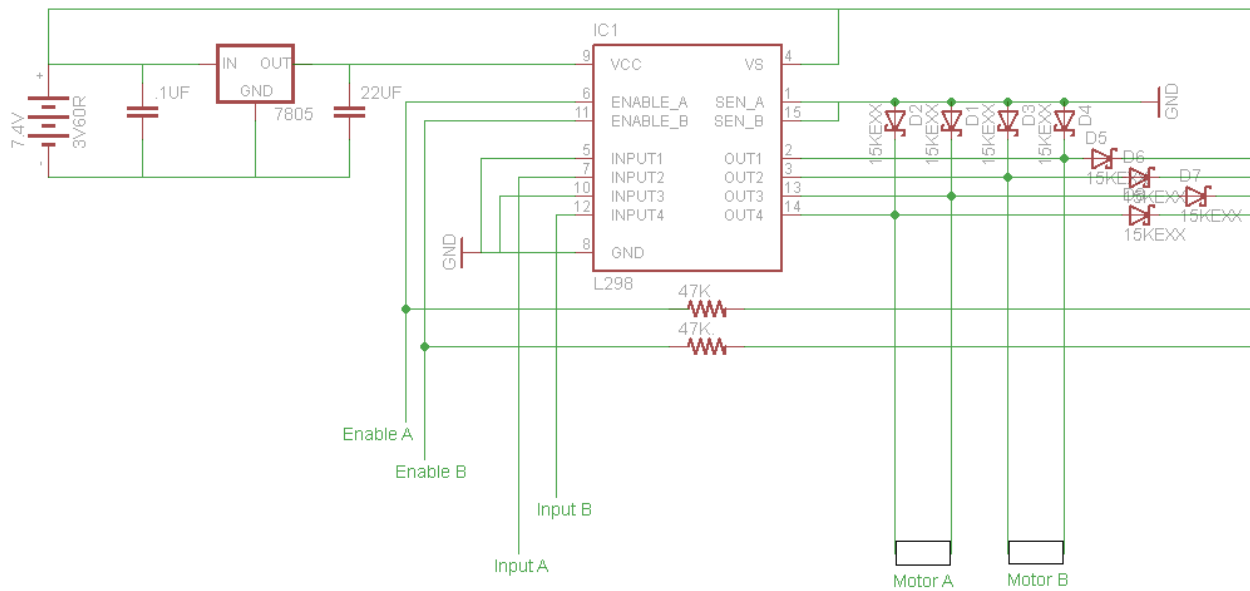


Figure 2: Motor Driver Circuit

Figure 2 shows our motor driver circuit, centered around the L298 dual full h-bridge. The components of the circuit were chosen based around manufacturer-recommended circuits, and availability of the parts in the parts shop. Parts chosen were taken either as specified, or at a higher rating, if the exact part was not available.

For instance, the schottky diode recommended was the 1N5818; we've chosen to replace that with the 1N5819, which is available at the parts shop, and exceeds the specifications of the 1N5818.

Similar substitutions may have been made for other components.

This circuit also contains our power source and voltage regulator. We will run the motors unregulated off of the battery, and we will use a 7805 voltage regulator to supply 5V to the rest of the circuit.

Figure 3 shows the PIC which will be responsible for controlling the drive motors, along with the RS232 communication circuit which will be used by both PICs. A MAX232 chip will be used to scale the 10V serial output from the BeagleBoard to 5V logic, which will then be useable by the PICs.

Figure 4 shows the second PIC, which will provide the PWM output to the servo motor. We need to use two PICs because we are using one PWM output for each motor, and each PIC has two PWM modules. This PIC will also similarly be connected to the RS232 communication circuit as in figure 3.

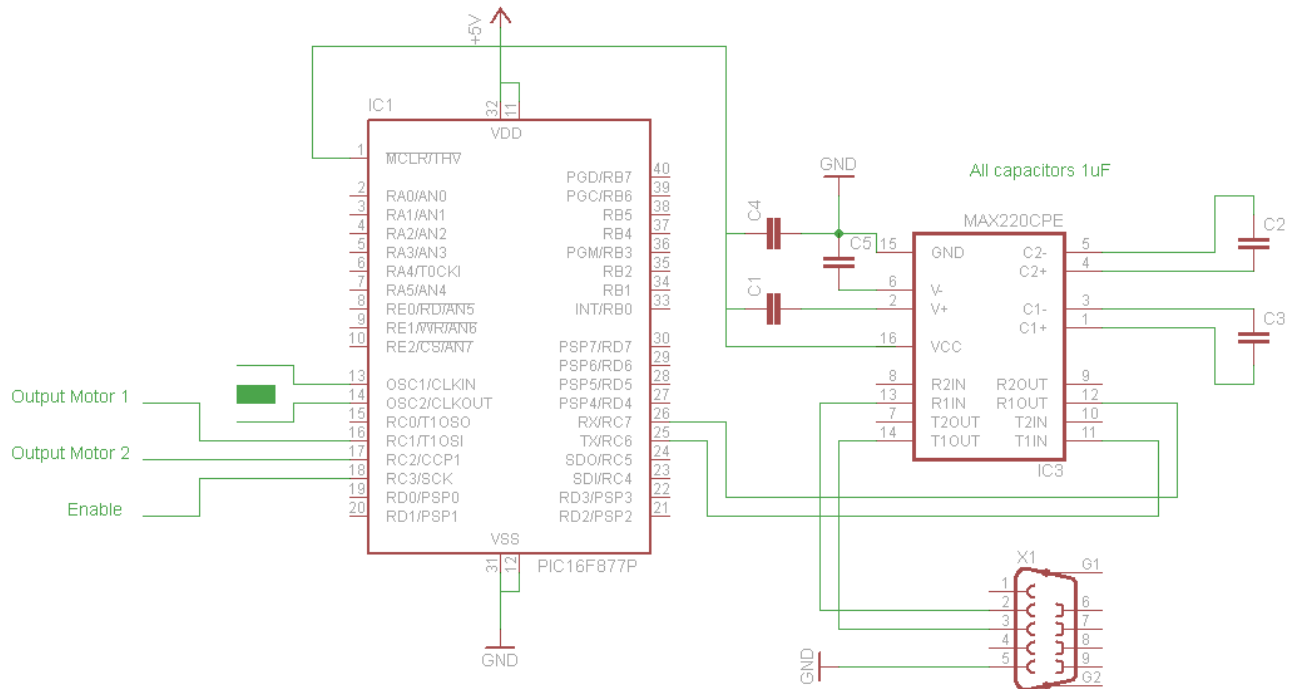


Figure 3: PIC and RS232 converter

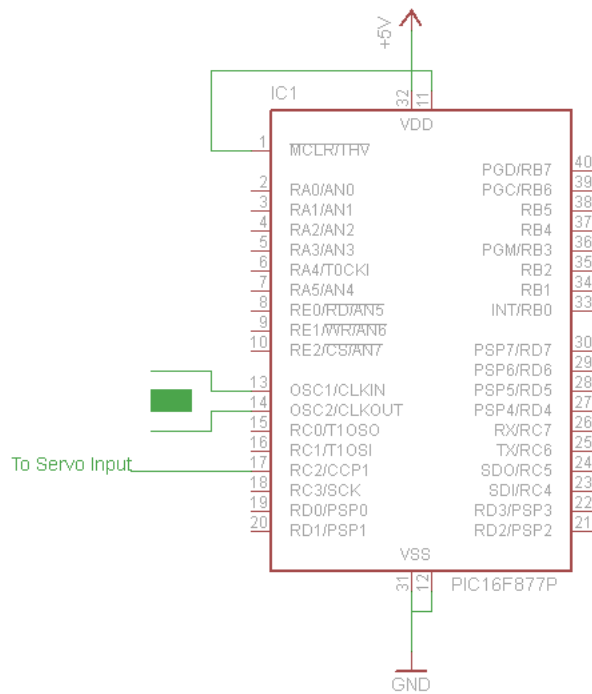


Figure 4: PIC Servo Controller

2.3 Image Processing Flowchart

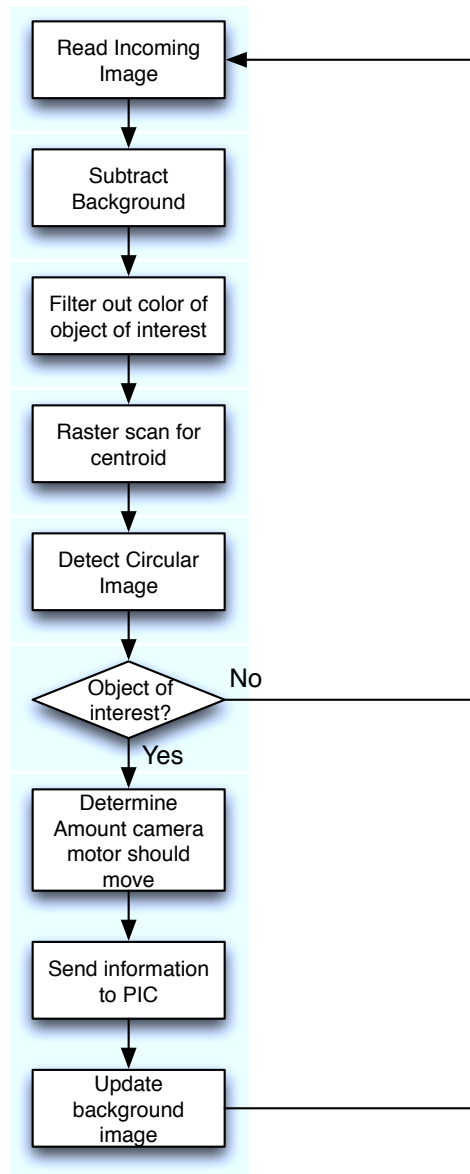


Figure 5: Main image processing routine flowchart

2.4 Image Processing Flowchart – Details

1. Read Incoming Image: The image is passed in from the camera. The passed in image will be converted to type double and grayscale version of it will be copied. Only during this stage will incoming frames from the camera be accepted. Our aim is to accept every 2nd frame from the camera and to do this, our code needs to run in under 66 ms.

2. BG Subtract: Due to the time constraint on our code, we are using BG Subtract instead of cross correlation to compute correlation as BG Subtract has a cost of N while cross correlation has a minimum cost of N^2 . BG Subtract simply takes the incoming gray image and subtracts, pixel by pixel the gray image of the saved background.
3. Filter out color: While BG Subtract is removing out the background, the incoming color image will have all its colors that closely match our object of interest highlighted. This is to provide further segmentation and breakdown in order to detect our wanted object. The two masks are then overlaid to produce a final mask showing a possible object of interest.
4. Raster Scan for Centroids: The mask is then placed through two raster scans in order for us to isolate each individual object and find their centroids for circular detection.
5. Detect Circular Image: As our object is a ball, the objects we have segmented from our image are checked for circular properties. This is done through the use of the constant radius property of a circle.
6. DECISION: Object of Interest? If a circle has been detected, we have found our object of interest so we continue. Otherwise the process is restarted.
7. Determine amount camera motor should move: Once we have the location of an object, we need to control our camera. The distance off center is processed through a second order control system. This is so that the camera movement is as smooth as possible which will then in turn help our image processing.
8. Pass Distance to Camera Motor: Once calculated, the amount of movement is given to the motor for the camera to start moving. This movement will be limited by the strength of the resolving power of the camera as fast movement will blur our images and destroy image processing. The movement must not halt other processes.
9. Update BG: The BG is then updated with the new background minus the objects that were detected.

2.5 PIC

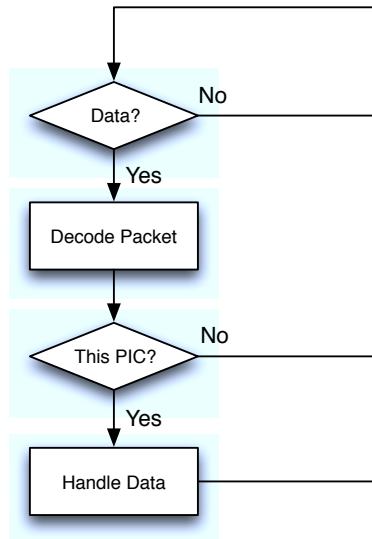


Figure 6: PIC information retrieval and update routine

The PICs will be responsible for sending out the appropriate control signals to the motors based on information it receives from the main program running on the BeagleBoard. This simplifies the program logic of the PIC; as seen in figure 6, it basically sits in a loop waiting for data on the USART; once it receives it, it updates the PWM output frequency based on information received in the packet.

Figure 7 shows the packet format used in communication between the BeagleBoard and the PIC. PID is the PIC ID – the unique identifier which selects either the first PIC or the second PIC. Both PICs receive the same set of packets, and will drop any packets which do not start with their PID. M identifies which motor the PWM rate should be set for – it is ignored on the PIC which controls the servo. Then, we have 5 bits of data for the PWM, and a terminating bit which identifies which of the 5 bits we are setting.

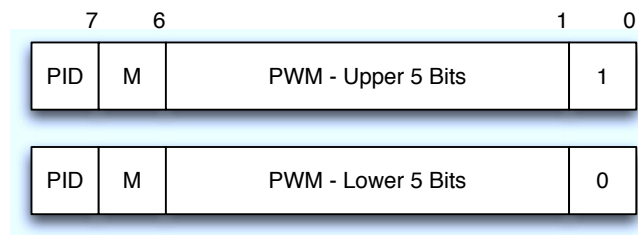


Figure 7: PIC packet format

2.6 Specifications

2.6.1 Camera Control

For our camera motors, we want it to be able to respond to the current position of the object of interest with relation to the center of the image and reposition itself quickly and with little overshoot while maintaining the stability of the motor (i.e. not overworking it). Thus, we will use a feedback control system, as seen in figure 8.

As the camera has an fps of 30, camera control is limited to having a maximum response time of 33 ms. So that we have room for error, the aim is to cut this response time down to 30 ms so that we don't have overlapped signals from one frame processing period to the next. Additionally, it must meet the requirement of little overshoot. Too much overshoot will result in a jittery configuration where the camera is required to correct errors from the previous iteration. For this, an overshoot no more than 15% is allowed.

Thus, in order to reach the required specs of 30 ms rise time and 15% overshoot, a second order feedback system is required. The position function is the current angular displacement from the center the ball is currently. This will be calculated based on the pixel distance from the center and the distance of the ball from the car (calculated through the size of the ball). The velocity function will come from the difference in the ball's angular velocity (measured from one frame to the next) and the camera's angular velocity in the world frame. The camera's angular velocity will then be equal to the camera's angular velocity in the car frame + the car's angular velocity in the world frame.

In summary:

$$x = \text{angular distance from center of image}$$

$$\dot{x} = \text{angular speed of ball} - \text{angular speed of camera} - \text{angular speed of car}$$

Using Franklin and Powells equations for feedback control: $\omega_n \geq \frac{1.8}{t_r} = 60$ and $\zeta \geq 0.5$ for a 15% overshoot.

The transfer function for a DC motor is:

$$\frac{K}{s(Js + B)}$$

Given this, we should be able to solve for the parameters we need for our control system. Since we don't have all of the characteristics of our motor, we will need to find them, plug them in, and then find the equation for our control system.

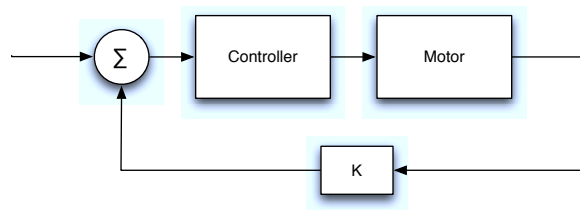


Figure 8: Feedback control system

This system will then be implemented digitally on the BeagleBoard, and will send control signals to the PIC, which will in turn control the motors. Through this, we will be able to control our camera to point at our desired object with a fast and stable response.

2.6.2 Car Control

We need the above motor to track the object because simply using the car to follow it wouldn't be quick enough, although we can still follow the object at a reasonable rate, as demonstrated below.

Once the camera has sought out the object, we would like the car to align itself with the object and follow it. We would like the car to continue following the object as it moves, regardless of any turns the object might make.

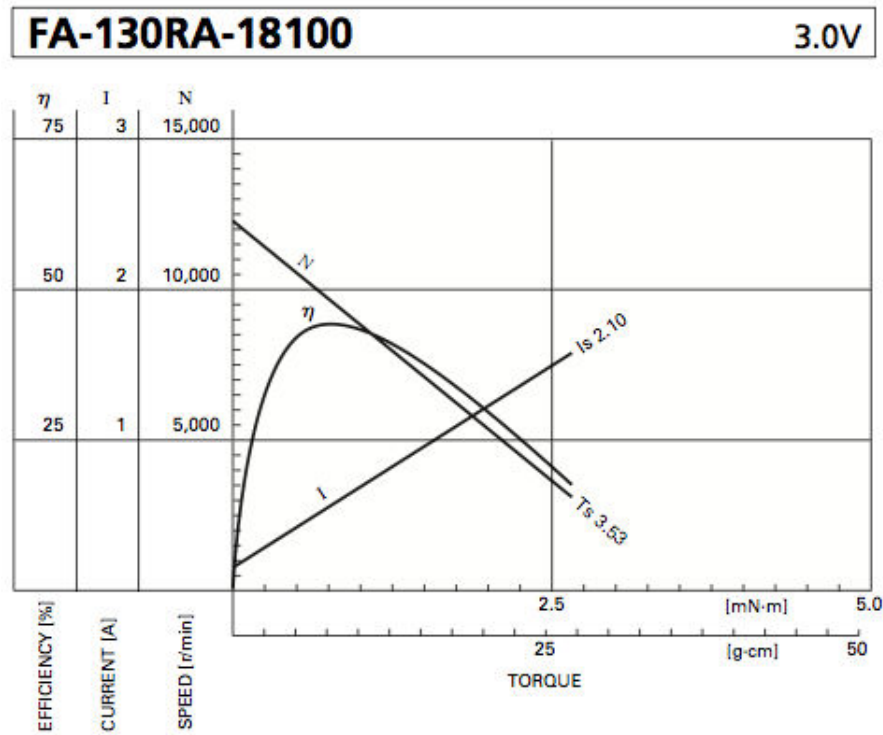


Figure 9: Motor performance curves for the similar FA-130RA-18100

As seen in figure 9¹, under load, our motors² will run at approximately 7,500 rpm, ungeared. With a gear ratio of 58:1, this will give us 129 rpm. Our wheels will be 1.8cm in radius, so from that we can calculate our maximum speed:

$$\begin{aligned}
 \omega_{max} &= 129rpm \times \frac{1min}{60sec} \times 2\pi \\
 &= 13.5 \frac{rad}{s} \\
 v_{max} &= \omega \times 2\pi r \times \frac{1}{2\pi} \\
 &= 24.3 \frac{cm}{s}
 \end{aligned}$$

¹Source: Mabuchi motor FA-130 (#18100) data sheet

²This motor is used as an example; our motors will be slightly different, running under slightly different conditions. These calculations will need to be redone once we have profiled the motors.

So, now we have our per-wheel maximum speed. This is the maximum speed that we can have each wheel rotate, and thus the maximum speed we can move forward. We can scale this back by about 50%, and we'll have 12 cm/s as our normal speed.

We do this so that we can move and make turns at the same time. In order to turn, we'll need one wheel to move faster than the other. Our maximum angle θ we can turn in time Δt is defined by the relationship:

$$\omega_{max} = \frac{\frac{\theta}{\Delta t} L}{r}$$

With r being the radius of our wheel and L being the length of the shaft between the wheels.

This is for one wheel stationary, the other moving – in other words, for the car to rotate in a tight circle. If we'd like to move at our normal speed of 12 cm/s and turn at the same time, we slightly modify the equation:

$$\omega_{max} = \frac{12}{r} + \frac{\frac{\theta}{\Delta t} L}{r}$$

Now, if we substitute in our maximum speed, the time between frames on the camera for Δt , and the length of our shaft, 10cm, into the equation, we can solve for the maximum angle that we can turn between every camera frame.

$$\begin{aligned} \omega_{max} &= \frac{12}{r} + \frac{\frac{\theta}{\Delta t} L}{r} \\ 13.5 &= \frac{12}{1.8} + \frac{\frac{\theta}{0.033} 10}{1.8} \\ \theta &= 0.04059rad = 2.32^\circ \end{aligned}$$

A cheap servo motor can turn 60° in about 120ms, or 16.5° in one camera frame. Thus, in order to be able to optimally center in on the object, we need to use the combination of the servo and the car turning.

For this, we will use PWM output from the PIC to keep the motor at the desired speed. The PIC has a 10-bit PWM duty cycle precision, which is more than enough for our needs. With a simple feedback control system relying on the offset of the servo from dead center, we will be able to tell the motors to turn by more or less, depending on the angular offset.

2.7 Performance Requirements

For this project, we require that our robotic car meet the following requirements:

2.7.1 Basic requirements

- Be able to isolate an object of interest within:
 - backgrounds with a normal room setting (i.e. not specifically built for extra contrast)
 - distances of at least 5 meters away from the camera
- Be able to track objects that:
 - are at most 15 cm x 15 cm x 15 cm in size (bare minimum isolation capability)
 - are moving at a speed of at least 10 cm/second
- Maintain tracking of object for at least 2 minutes.

2.7.2 Advanced goals

- Detect and remember where an object disappeared and travel to that location to try and reacquire object.
- Automatically turn on detecting object of interest entering camera window.
- Avoid obstacles while chasing down and tracking an object.

3 Verification

3.1 Testing Procedures

In order to test our entire project, the best method was to first break up the testing procedures into various independent critical sections.

3.1.1 Object Recognition Software

The essential piece of our project is the object recognition software. The software must be able to identify our object of interest from a video image and isolate it. This testing process will be broken into two elements within the software, object isolation and object tracking.

1. Object isolation: This software element is simply tested by printing out the original image and the processed image with eliminated background. We want our software to be able to detect an object of interest and be able to isolate it for objects of size 15 cm x 15 cm x 15 cm at a distance of at least 5 meters.
2. Object Tracking: This part of the software must be able to identify the current location in real world terms of the object. This will just be tested by seeing how accurate the code can return the object's current world coordinates given its isolated image. We want this to an accuracy of 50 cm laterally and about 1 meter longitudinally, and would like it to work for an object moving at a speed of at least 10 cm/second

Graphs can then be made showing the region of images that can be isolated (based on image size and difference from background) and the region of images that can be tracked (based on speed of image).

3.1.2 Camera Navigation

The next important element to our project is its ability to direct the car's camera to a place an object at the center of the camera. This can be isolated from the object recognition part by manually inputting an imaginary object location and letting the camera's program adjust to that location. We will then physically measure, with ruler and protractor, where the camera moved to, and find the difference between where the camera moved to and the expected location.

3.1.3 Centering of Camera

Once the camera has captured the direction of the object, the car must turn and move in that direction. This can be simply tested by having the camera have an initial offset direction. We will then fix the camera direction with relation to the world, and the car will slowly turn so that it will

face in the same way as the camera. When it's done, we will again physically measure the result and compare it to the expected position of the car.

3.1.4 PIC Microcontroller

The PIC needs to be able to receive control signals from the BeagleBoard, and output from the PWM the desired frequency. To test this, we will hook up the PIC to the BeagleBoard, and then hook up the PWM outputs to an oscilloscope. We will then send control signals, and measure the corresponding duty cycle of the output on the oscilloscope.

3.1.5 Motor Driver

The motor driver simply needs to take the logic-level PWM output and use it to drive the motors. To test this, we will use a function generator to generate a signal to simulate the PWM, and run the motors. We will measure the rate of rotation of the motors by timing their rotations³, and compare them to the calculated values.

3.1.6 Overall Operation

Once each individual element has been tested and verified as working, we will then test the full program's functionality, as a whole. The main reason is this project also has a lot of communication from one element to another (mainly feedback control). Without testing the project as a whole, we will not be able to ascertain that this communication is working. So as to gradually see and isolate any complications, the environment for testing will be increased incrementally in complexity until we have reached the same environment as an ordinary room.

3.2 Tolerance Analysis

One major area of concern for tolerance is the processing speed of the program from image acquisition to car/camera control signals outputted. As this project is done in real time, we need for the delay from image acquisition to the sending of control signals to be as short as possible.

In order for our car to maintain the tracking and movement of an object moving at a speed of 1 meter per second as specified in our design criteria, we require our program to at least have at most 33 ms processing speed or we risk running into the problem of the object running off image. We will thus test our algorithm to make sure the runtime fits within these bounds.

Secondly, we need to be able to actually follow our object. Thus the car must have a minimum speed of 10 cm/s to allow it to keep up with the object of interest. In addition to this, speed control would be needed so that we don't pass the object. The motor controller needs to be able to match the object of interest's speed to a high degree of accuracy (less than 0.1 m/s of error). This can mostly be done through feedback control inside the program to try and maintain a constant distance between the car and the object.

³This will only really work for slower speeds, but should give us a good idea of their operation. For faster speeds, it will be enough to check that the speed increases as the PWM duty cycle changes

4 Cost and Schedule

4.1 Cost Analysis

- Labor

	Hourly	Hours	Total	×2.5
Richard	\$35	200	\$7000	\$17,500
Brian	\$35	200	\$7000	\$17,500
Ruben	\$35	200	\$7000	\$17,500
Total			\$21,000	\$52,500

- Parts

Item	Quantity	Cost
PIC 16F877A	2	10.00
Beagleboard Rev. C4	1	150.00
SD Card	1	10.00
Beagleboard Cables/Accessories	1	10.00
Logitech C200 Webcam	1	26.00
Servo motor	1	7.00
Robot chassis	1	10.00
Twin-motor gearbox	1	10.00
Brushed DC Motor, 6V	4	7.16
Battery	2	20.00
L298 H-Bridge	3	8.00
7805 5 volt regulator	2	2.00
MAX232	1	1.00
Misc. resistor/cap/semiconductors	1	10.00
Total		\$281.16

- Grand Total

$$\text{Labor} + \text{Parts} = 52,500.00 + 281.16 = \$52,781.16$$

4.2 Schedule

Week	Task	Responsibility
2/15	Start schematics	Rich
	Research all parts	Brian
	Begin planning motion-detection algorithm	Ruben
2/22	Make sure all parts ordered	Rich
	Finish schematics	Brian
	Motion detection simulations	Ruben
3/1	Get BeagleBoard running and start programming PIC	Rich
	Implement C code for motion detection	Ruben
	Implement and test motor control circuits	Brian
3/8	Test motor control circuits and control algorithm	Brian
	Integrate control signals to center camera on target	Ruben
	Make sure PIC and BeagleBoard working together; integrate with motors	Rich
3/15	Refine motor controls and circuits	Brian
	Write interface code between motion detection and motors	Rich
	Test motion detection and algorithm outputs	Ruben
3/29	Refine and test interface code	Rich
	Integrate motion detection with car movement	Ruben
	Help test and integrate motion detection	Brian
4/5	Test entire system, individually and together	Everyone
4/12	Get PCB made	Brian
	Attempt to implement obstacle avoidance	Rich
	Work out edge cases, attempt to implement history-based tracking	Ruben
4/19	Individual reports, debug and make sure final system works	Everyone
4/26	Demo	Everyone
5/3	Final Papers	Everyone