04/22/08
Student Name: Barry Solomon
TAs : Adam Barnett
Mike Pridgen
Sara Keen

Rack Attack

EEL 5666: Intelligent Machines Design Laboratory,

University of Florida,

Drs. A. Antonio Arroyo and E. M. Schwartz, ECE

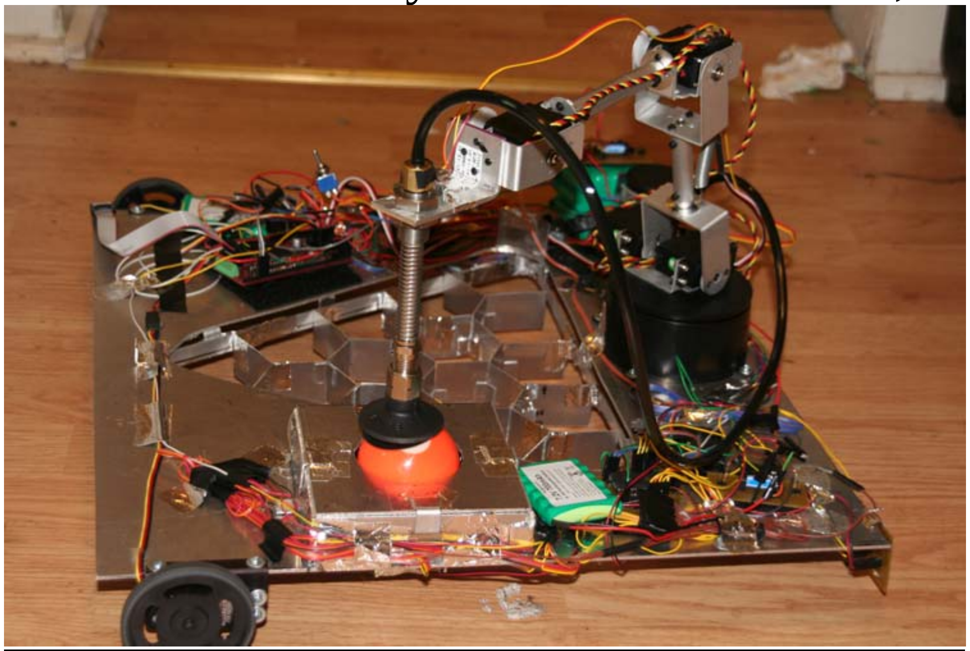

Table of Contents

Abstract

Rack Attack is an autonomous pool ball retrieval and identification robot. It has a four degree of freedom robotic arm with a vacuum cup on the end that it uses to pick up pool balls and move them. Rack Attack tries to determine both a balls color, and whether or not it is a stripe or a solid. Once color and type have been determined, Rack Attack stores each ball in a partitioned onboard rack that has a specific spot for each ball. Rack attack can also follow walls, and may one day be able to navigate a pool table. This paper will describe how Rack Attack works.

Executive Summary

Rack Attack was constructed in the spring of 2008 for IMDL. Its original intent was to be an autonomous robot that once placed upon a pool table, would navigate its way along the walls of the table, stopping at each pocket, retrieving the balls, identifying each ball, and then placing each ball in it's appropriate position within an onboard rack.

However, the navigation, and the need to identify the balls position within the pocket, proved more difficult that had first been assumed. Eventually challenges on both of these fronts proved insurmountable and the actual functionally of Rack Attack was scaled back. It is able to follow a wall, and stop if that wall stops, but it cannot navigate a pool table. It is able to move its arm to predefined positions, or positions adjusted by the robots distance from it's left side to the wall on that side. The robot can also pick up pool balls, identify them, and put them in their correct rack positions, it just has to know what height the balls are at, it cannot sense that.

Rack Attack uses bump sensors, infrared sensors, and cadmium sulfide (cds) cells to do its sensing. It has a total of seven servos on board, and a vacuum pump. Everything is controlled by an ATmega128 microcontroller on a Mavric IIB board. The whole setup also requires four separate batteries to run.

Introduction

After an intense game of pool, the last thing anybody wants to do is waddle around the table to pick up all those balls, move them to the front, and try to figure out which one goes where. I know I just want to sit down, relax, have a drink, and rest up for the next round. Sure you could hire an illegal immigrant to do it for you, but think of the legal implications. Wouldn't it be better if there was a robot that could do all this for you and make you the envy of every Joe on the block? Enter Rack Attack, the first (maybe?), fully functional (not really), fully autonomous, pool ball retrieving and racking robot.

Rack Attack can and will identify all colors of standard pool balls, and it can determine if they are a stripe or a solid. It can pick up pool balls with its robotic arm, and rack them in predefined positions. It can also drive along a wall, and stop if it comes to an opening like a pool pocket.

Rack Attack has also been special constructed to be gentle on your pool table. The end of the arm has a gentle silicone vacuum cup for retrieving the balls, and the drive wheels are plastic with rubber tires. Additionally the platform of the robot sits slightly above the railing of standard tables.   All this careful planning ensures that no metal, or no sharp edges will ever touch the table.

The rest of this paper will explain how Rack Attack does what it can. First how the system as a whole functions together will be discussed, and then all of the components will be explained individually.

Integrated System

Rack Attack is controlled by an ATmega128 microcontroller on a Mavric IIB development board purchased from BDMicro. The Mavric IIB board was recommended to the class, and it's capabilities were well suited for the needs of Rack Attack. The microcontroller board is interfaced with everything on the robot, all the sensors, the LCD screen, the servos, and three relay switches. Figure 1 below displays how the various components interact and communicate.

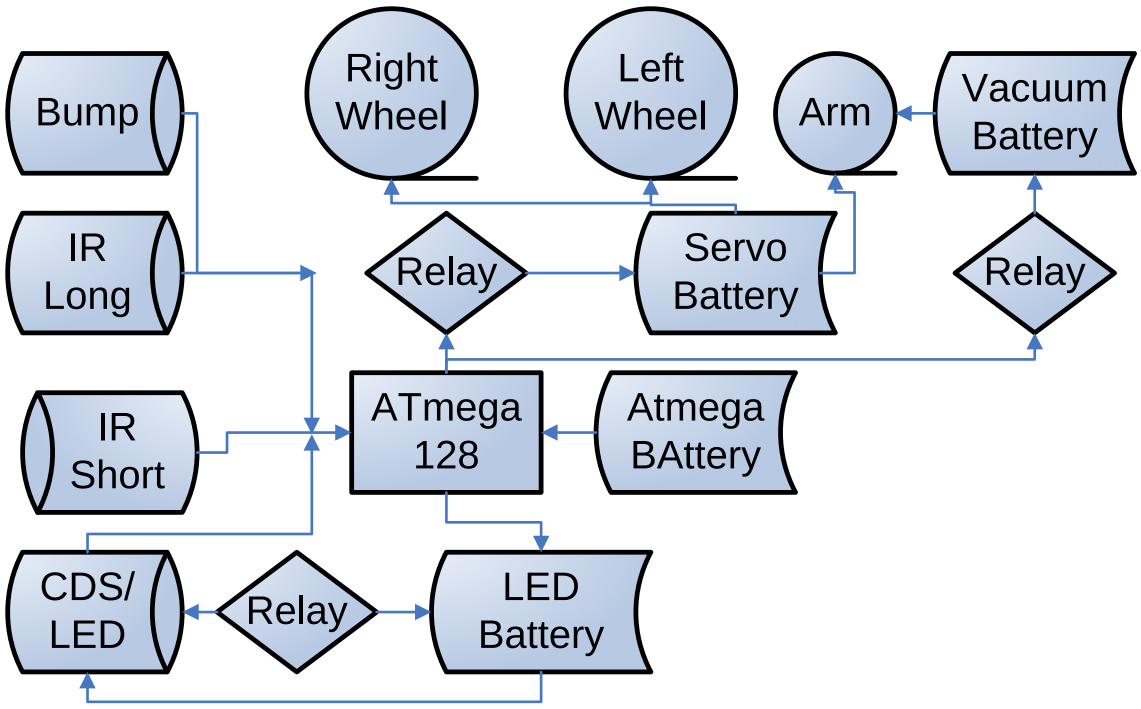

Figure 1

The basic operation of the robot is as follows (for the ideal case):

1) Turn on
2) Two short range IR facing left take distance measurements
3) One long range IR on the front and one on the back also take distance measurements.
4) These IR sensors are calibrated on startup
5) Using the side distance measurement, and some programmed constants, the arm moves to the pocket to extract the balls within.
   -The arm is powered by 5 servos.
6) Bump sensors on the end of the arm tell the robot when it has reached a ball

7) Then the relay for the vacuum pump battery is enabled, and the vacuum turns on so the ball can be lifted by the arm.
8) The ball is moved to a box that contains five LEDs each paired with a CDS cell. These five sensors shine 7 different colors of light at the ball and each take a reading for each color.
   -This information is used to determine what color the ball is.
9) The ball is moved to its position in the rack.
10) Then the arm goes back to the pocket to look for more balls.
11) If no more balls are left in the pocket, then the robot moves on to the next pocket.
12) The robot uses the front and back IR to tell it when to turn.

Mobile Platform

The Platform was constructed entirely out of aluminum since it needed to be sturdy, and light. There is a large triangular shaped hole cut out in the center where the balls are racked. Since this hole spans almost the entire width of the robot, aluminum was the best choice to provide rigidity, and to keep weight low. The platform sits on four wheels. Two up front that drive the robot (each has its own servo), and there are two castors in the back. The castors were placed a few inches from the edges of the robot to help the robot have good stiffness where the arm would be positioned (in the center). There is a partition in the racking space that snuggly fits all the balls. It was made out of aluminum as well, and it many walls were attached with aluminum brackets that were glued to partition panels. There is also a box made out of aluminum on the robot where the color sensors are housed. This was done to keep them away from the light so they could take accurate readings.

The IR sensors are mounted on the bottom of the robot. This is so they can see the walls of the pool table, since the platform sits higher than the rails do. There are also many less wires on the top, than on the bottom. Two bump sensors are on the back edge of the robot to tell it to stop when it is backing up to a pocket.

The platform was made square so it would be easy to deal with its geometry, but it was kept as small as possible, since it needs to be big to accommodate all the balls and the hardware. The arm was placed at the back middle so it could easily get to the pocket, the color sensors, and the rack, in a sequential type of movement.

Actuation

Rack Attack is driven by two servos, one in each of the front wheels. They are the same servos, and they have been modified to run 360 degrees continuously. They get their signals from the microcontroller in the form of PWM signals. They are driven by a separate battery from the board.. This is because I wanted to make sure there was enough power for the servos, and the microcontroller, and I didn't want the servos to overload the board with current. The servo battery is activated using a relay that is switched on from the board. One of the servos is rotating forward, and one is rotating backward, so it was difficult to find the matching PWM signals that made them both run a similar speeds. The servos don't move as fast as motors would have, but that is the reason I chose them. I wanted the robot to move slow and controlled.

There is also an arm, which is used to move the balls around. The arm has 5 servos, 4 degrees of freedom and one vacuum cup attached to a vacuum pump. These servos are also controlled by PWM signals sent from the board. The physical size of the arm was decided first based on necessity of function, and then based on the weight of the components and the weight of a pool ball, the various servos were given necessary torque ratings , so I knew what I needed to get to make it all work. I used servos all from the same company, HITEC, so that way I knew they could be controlled similarly and that their specifications would have been calculated similarly.

The arm has one servo at the base to rotate the arm. This servo is the weakest since it has the smallest load. The shoulder of the arm has two servos working in tandem, and these servos are high torque digital servo. The elbow has the same servo that the shoulder does. The wrist has a slightly weaker digital servo, but it is still very strong. The digital servos have about 150 degrees or rotation while the analog servo in the base can rotate a full 180 degrees.

The vacuum cup at the end of the arm is connected to a vacuum pump, that is powered by a 12VDC battery whose activation is controlled by another relay switch that is controlled by the microcontroller. The Vacuum pump is a swing piston type, and it has 300 mbar absolute ultimate vacuum, which is more than enough to lift a six ounce pool ball near sea level. The pump also has a 3.3 liter/minute vacuum rate which is plenty to make up for losses at the vacuum cup / ball interface. I have had some problems with the

12 volts causing the relay to get stuck in the on position, or the board not being strong enough to power a relay switch capable of activation the vacuum battery.

The arm is fully capable of lifting pool balls with ease, and battery life is no problem for the robot to retrieve all the balls. It was somewhat harder than I originally thought to get the arm and the wheels moving like I wanted, but lots of geometry and trial and error finally got them both working well.

All the arm servos pretty much move the same for a given change in the PWM signal. They may have different starting or ending pints, but they all move one degree for every 10 micro second change in PWM signal.   This was found by giving the servos a sequence of PWM signals that varied by .1 milliseconds, and measuring the angle at each point. The change was completely linear as can be see in the graph below.

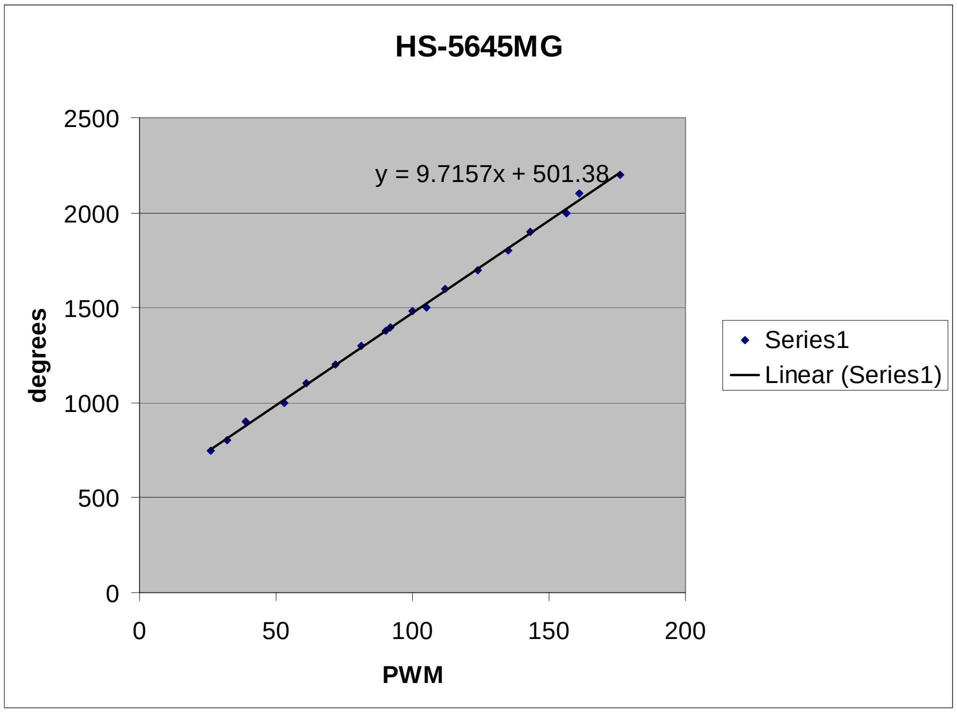

Figure 2

Sensors

IR

There are four Sharp IR sensors on the robot, one on the front, one on the rear, and two on the left side. The two on the left side have a closer operating range than the two on the front and the back. This was done since the robot is designed to keep its left side against the wall (a short distance), while the front and rear IR are required to record a much longer range.

The front IR sensor has the job of telling the robot the distance to the upcoming wall, while the rear sensor has the job of the telling the robot how far it is has traveled since the previous pocket. One IR sensor might be able to do this alone since the dimensions of a pool table are known, but not all pool tables are the same dimensions, and having two IR sensors gives the robot away to check itself. These distances will be used to determine the speed of the wheels, and to tell the robot when to begin a turn.

The two IR sensors on the left side of the robot will take distance readings at the front and rear of the robots side. These measurements will be used to keep the robot alongside the wall to its left while it moves between pockets, and to help the robot align itself to its left wall when completing a turn. The rear side IR sensor will also tell the robot when it is in position at the next pocket since its distance reading will be significantly higher than that of the front side IR.

One Problem with the IR sensors is that they work differently under different lighting conditions, and the two different IR sensors work differently as well. This can all be seen in the following table of analog to digital readings at different distances in two different lighting conditions, with the two sensors.

Dist (in)	Dark Long	Ambient Long	Dist (in)	Dark Short	Ambient Short
2	75	95	2	158	158
4	111	117	3	144	147
6	132	128	4	119	120
8	121	117	5	99	99
10	109	106	6	85	84
12	88	92	7	75	73
14	72	83	8	65	65
16	54	73	9	59	60
18	44	65	10	53	54
20	31	59	12	45	45
22	31	54	14	40	40
24	31	51	16	35	34
inf	31	34			

Figure 3

Therefore the robot will, at startup, take a reading from the side and rear IR sensors, while the robot is positioned in a corner of the table (so the distances will be known). It will use this reading to calibrate the sensors for the rest of its operation.

GP2Y0A21YK
(short distance)

GP2Y0A02YK
(long distance)

Figure 4

Bump

Positioned at the end of the robots arm will be a specialized bump sensor array.   The original design had this bump sensor being radial in nature, with four separate bump sensors, and has 12 protruding rods (three rods to a sensor). The twelve rods would hang down from the sensor in a circle evenly spaced. The sensor would then be moved slowly closer to the pocket. As the rods come in contact with a ball they would be moved upward, which would cause them to break their individual leg of their circuit, causing that bump circuit to change its voltage reading. Logic within the robot would have determined where the ball is based on which rods hit and known distances to the rods.

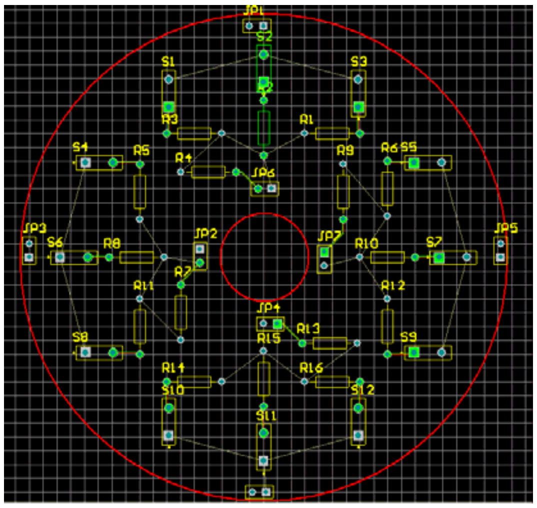

Figure 5

However, this setup never worked right, because all twelve rods would never maintain good contact with their contact points when then were not touching a ball. Many different solutions were tried, but nothing ever worked right. I still think this or some version of it is the best idea, but I didn't not have the materials or the time to get the materials to do this better. In the end what was done, was a small bump sensor was placed on the bracket at the top of the vacuum cup assembly. This way, when the assembly hits a ball and moves down its own spring, the bump switch comes out of contact. This enables the root to know at that it has reached the ball and that it should stop moving down. This does not however allow the arm to find the ball if it is off center.

There are also two simple bump sensors at the rear of the robot. These will let the robot know if it has reached a wall when backing up. Figure 5 below shows the bump sensors on the robot, and how their circuits work.

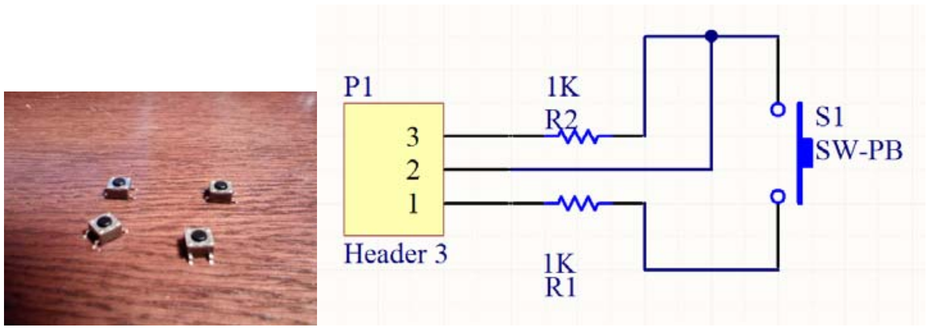

Figure 6

Light

To determine the color of the balls, cadmium sulfide cells (CDS cells) are paired in tandem with light emitting diodes (LEDs). The CDS cell changes resistance in response to the amount of light it sees, and the LEDs will be this light source. Depending on the color of the ball, and the frequency of light from the LED, different intensities of light will be reflected back towards the CDS cell from the ball. This will allow the robot to determine the color of the balls. Below is a table of tested values using LEDs that produce seven different colors.

Setup **Ball Color**	Red Light	Blue Light	Yellow Light	Aqua Light	Puple Light	White Light	Green Light
White	159	222	154	215	155	153	216
Yellow	163	245	153	226	159	161	215
Blue	225	234	223	237	217	220	241
Green	218	245	214	237	217	220	241
Orange	162	162	147	148	148	146	163
Black	233	247	231	247	229	230	253
Purple	220	241	219	239	212	214	250
Burgundy	198	244	200	240	200	200	249
Red	198	244	200	240	200	200	249
Nothing	254	254	254	254	254	254	254

Figure 7

Many different resisters were tried in the circuit until ones that gave the most spread in the readings without introducing too much noise were found. The resisters with the LEDs were 97 ohms, while the resisters with the CDS cells were 4.7K ohms.

It is necessary to keep out ambient light, so location does not affect the readings from the CDS cells. Therefore, the ball is transferred to a sensing chamber by the arm after retrieval from a pocket. The chamber will house 5 CDS/LED sensors, there will be four around the sides of the ball, and one underneath it. This chamber will allow half of the ball to be investigated by the sensors, while the other half protrudes above the chamber blocking light from coming in. The robot has been programmed to cycle through all seven colors of the LEDs and take readings for each. Then every ball has a set of 2 to 4 logical parameter that set it aside from the other balls for certain colors. The balls will also be identified as stripes or solids based on how many similar reading they get from the five sensors. Three or more matches means it is a solid.

These LEDs have three pins on them, one for power, one for ground, and one that switches it to the next color. The switch lead needs to be connected to ground to cycle to the next color (it actually goes color-off-color-off…). A relay switch is used to turn ground on and off that third pin. One problem that arose was that these LEDs did not function correctly when hooked up the board. They would only cycle through three colors, but they had worked fine in testing. I finally decided that ground must have been acting funny on the board, so I hooked the LEDs to their own battery. This worked and they now cycle through all seven colors, but the voltage of this new battery is higher, and the lights may not be shining at he same frequencies as before, so their reading may not be calibrated correctly now.

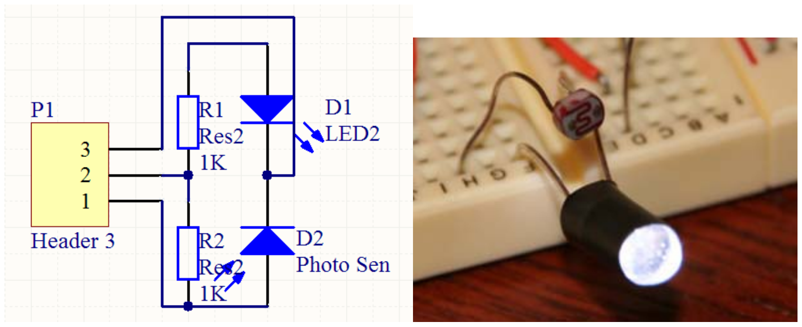

Figure 8

Behaviors

The Robot was supposed to wall follow along the rails of the pool table. This proved difficult because since the robot is so long. I first the robot speeding up one wheel when there was a discrepancy between the side IR, but by the time the other side would catch up, it would be going to fast and Rack Attack would over correct. I fixed this by only having one wheel ever change speeds instead of both. So one wheel either speeds up or slows down depending on what is needed. I also put a delay in between changes, and I said that no changes should occur if the wheels the robot is making forward progress towards correcting itself. This seemed to stop the over correcting problem, but the robot does tend to lean to the left still.

Below is a drawing of the projected path of the robot when navigating the table. Basically the robot drives, and till it gets close to the next wall and then turns before the wall and back up to the pocket where it stops so that the back left corner of the robot is positioned at the pocket.

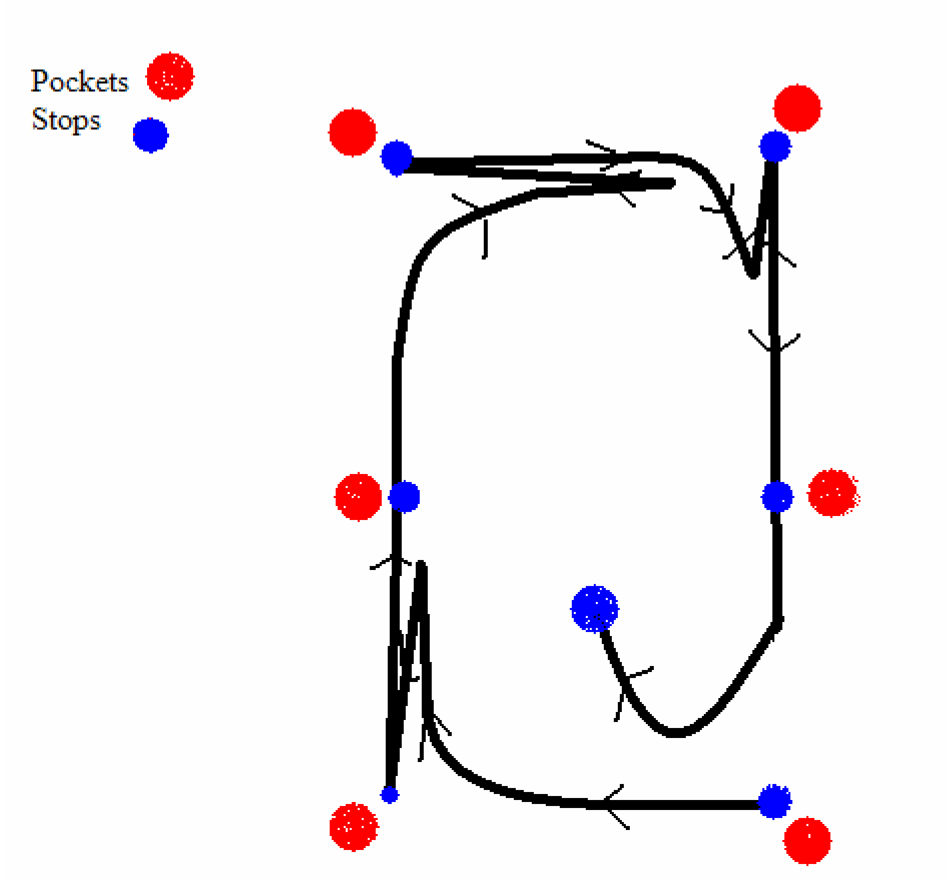

Figure 9

The side IR are supposed to also tell the robot when it has reached a side pocket. First when the front side IR reaches the pocket its value would drop suddenly, and the robot will slow down. Then once the robot waits for the back side IR to drop suddenly in value, and then as soon as it starts raising its value (just passed the halfway point of the pocket) the robot would stop, and the arm would be lined up perfectly with the hole. Problems arise in the form of the robot not being aligned right with hole, from the robot being not straight, or not stopping when it should.

Conclusion

Rack attack can move its arm to any spot defined by a radial coordinate system. The arm can go to the pocket (assuming he robot is parallel to the wall), pick up the pool ball, drop it in the color sensor box, and move it to the rack. The color sensor box can reliably find the color of any ball, but it does have some trouble with stripe solid identification. It sometimes mistakes solids for stripes. The robot can wall follow well, but it does not accurately stop itself at the pocket. The IR sensors should be enough to navigate the table, but much more time is needed to work on the programming. Things just don't happen as you think they will when you are programming.

Also the bump sensor on the end of the arm does not work well, and needs to be redesigned. In retrospect, it might have been better to attach some sort of camera to the end of the arm to find the balls in the pocket, but this would have added to the weight. So I am still not sure what would work best. An IR on the end could tell you how far away the balls are, and that might be enough, but the vacuum cup might get in the way so this could be a problem. I still think the customized bump sensor would work best, but it needs to have springs incorporated somehow, because gravity is not good enough to hold the rods down when they are not in contact.

I want to keep working on the robot though, even after the class is over, and one day it will do everything we hoped it would.

Appendices