The Caddy

Palani P. Rathinasamy

EEL 5666

Intelligent Machines Design Laboratory

Department of Electrical and Computer Engineering

University of Florida

Date: April 22, 2003

TA: Uriel Rodriguez

Jason Plew

Instructor: A. A. Arroyo

Table of Contents

Abstract

The golf caddy is an autonomous robot that will assist a golfer at the driving range. One of the most frustrating things about going to a range to hit balls it the fact that the balls have to be teed after each previous ball has been hit. After hitting a few hundred balls, the constant bending is extremely painful to the lower back. This is where the golf caddy will come into play. The robot senses when a golfer has driven the ball and then places a new ball on the tee and moves away. It then waits until the ball has been driven and places a new ball on the tee. Attached to the caddy is a bucket allowing it to hold numerous amounts of balls

Executive Summary

The golf caddy robot idea came about in the first weeks of the semester while chipping balls in the backyard of my apartment. In the beginning design stages, the sensors, microcontroller and body of the robot were developed in the IMDL lab.

The body for the robot consists of two parallel circular pieces of wood. The upper piece of wood holds the balls while the bottom one houses the delivery mechanism, circuitry, motors and various sensors.

In order to prevent the robot from running into objects as it proceeds to the golf tee, various sensors have been installed. Two sonar sensors on the front can detect objects up to 3 meters away. Four bump sensors attached directly to the surface of the robot will act as last chance collision avoidance in the event the sonar does not

To find the tee as well as to find a path to sit and wait while the ball is hit, the robot will follow a line on the ground. In order to follow the line, two IR sensors and a CdS cell were used. The two IR sensors were used on the outside of the CdS cell to determine if the robot had strayed in either direction from the line.

In order to determine if a ball had been hit off the tee, a CdS cell was placed inside the tee and attached to a RF transmitter. When the ball is hit off the tee, the RF transmitter will send a signal to the microcontroller notifying it of this condition.

Introduction

In the past quarter century, a huge revolution in the design of robots has occurred throughout the world. From the needs of industry for robots, robots also are starting to play a crucial role in the domestic lives. Autonomous cars as well robots that traverse far away planets have already been made. As humanity grew from the beginning, it created tools to help it deal with everyday tasks. Robots are the future 'tool'.

The objective of this project was to build a simple robot that would place a golf ball on a tee and determine if another ball had to be placed on the tee. After the idea of this project was conceived, the first step involved ordering the microcontroller which would control all the various sensors and motors.

The next step was to design the mold for which the robot would be housed on using Autocad. The T-Tech machine in lab was used to cut out the wood. The sensors and motors were then ordered and mounted onto the platform. Next, the process involved writing code to integrate the microcontroller with the sensors and motors. The final step involved updating and debugging the code and making slight modifications to the hardware as deemed necessary.

Integrated System

Brain

The brain of the robot is the Atmel mega 103 microcontroller on an Olimex AVR-H103 development board. The chip is no longer developed by Atmel as they have switched to the Atmel mega 128. All the code that was made for the mega103 works with the mega128 microcontroller. Once of the reasons for choosing this chip was the

enormous abundance of I/O pins (48) as well as the 8 A/D lines. The major drawback to this chip that was realized after purchasing it was the limited amount of Output Compare Systems (3) and Input Capture Systems (1). Luckily in this project, there were only three motors used and the output compares sufficed. However, there were two sonar modules being used that required input capture. Since there was only one input capture system, input pins were used to time the reflected wave. In the future, if more motors or sonar systems were used, I would not use this chip and would seek a chip with more Output Compare and Input Capture systems.

Features of the chip:

-Pulse Width Modulator: 4 ch(s)

-External data memory interface (64kB)

-General Purpose Registers (Accumulators): 32

-I/O Pins: 48

-Analog-to-Digital Converter (10-bit): 8 ch(s)

Pricing Information

Cost per chip: $30

Olimex Corporation (http://www.olimex.com)

Mobile Platform

Objective #1- The mobile platform for this robot must be very agile and able to pivot easily around the middle as it will be line following from underneath. It must also be

able to handle obstacles in its path and not get stuck on them. Also, the platform should be able to hold a bucket on the top to hold the golf balls. The bucket's base diameter is nine inches.

Design- My original solution to the problem was to design a one story platform that would house both the ball dropping mechanism as well as the bucket to hold the balls. I ran into problems in that I would have to make the platform extremely wide in order to handle both the bucket and the ball delivery system (as the systems would have to be placed away from each other). The solution to this was to create two pieces of circular wood with 11 inch diameters on the T-Tech machine and to place them parallel to each other. The top piece of wood would hold the bucket and the balls while the bottom piece would hold the microcontroller, delivery system, motors and sensors.

Objective #2- Design a ball delivery system that will place a ball on a tee. This system must be fairly accurate and drop the ball at the same place every time. It must also ensure that multiple balls are not dropped at the same time.

Design- My initial solution to this problem was to design a ramp system. This system would stop the ball on the ramp and wait till it had to be dropped. Some of the problems with this system was that two extra servo's were to be mounted at an angle to stop the balls. Also, when the robot moved forward or backwards, the balls would easily come undone and bounce out of the machine. After scrapping this idea, I decided to use a gumball machine design in balls would drop into a catch that would rotate and deliver the ball. While a ball was in the system, it would prevent other balls from falling into the catch.

Actuation

Motion

Objective – Since the robot will have be delivering golf balls, the weight will be changing constantly. Fully filled, the robot will weigh 6 pounds and can weigh as little as 2 pounds. Any motors that are used must be able to handle the full range of the load as well as deliver ample torque to heavier weights.

Solution- My initial solution to this problem was to use stepper motors as they are extremely precise. This preciseness is extremely important as the robot must be fairly close to the tee in order to deliver the ball. The precision is because inside the stepper motors are poles that the motor can be accurately stopped on (allowing small angles of rotation to be achieved). The drawback to stepper motors however is that it is extremely difficult to control without a motor driver which can be expensive. After trying stepper motors for a few weeks, I decided to switch to hacked servo motors which are much easier to use because they are basically gear motors. The servo motors used were the Cirrus CS-80MG.

Problems- My initial hacking method turned out to be fatal to my microcontroller on three occasions. While hacking the servo, I failed to calibrate the potentiometer making the servo stay off center and the pins floating. While I didn't think that this would be a problem, it proved havoc on my microcontrollers and burned right through them. Three servo's later, and after talking to Uriel Rodriguez, I decided to strip the internal circuitry of the servo and treat them as strictly gear motors. To drive these high torque motors and prevent feedback current from destroying the rest of the circuitry, two Texas Instruments SN754410NE driver chips were used.

Motor Features: Cirrus CS-80MG

-129.8 oz-in torque

-0.25 sec @ 60 deg

- Metal Gears

Pricing Information

Cost per motor: $22.99

Hobby People (http://www.hobbypeople.net/)

Wheels

The wheels were purchased online from Acroname to fit the specific servo's that were being used. They are 2-3/4'' across and 5/16'' thick.

Pricing Information

Cost per wheel: $3.50

Acroname Inc. (http://www.acroname.com)

Sensors

1. Obstacle Avoidance
2. Line Following
3. Ball Detection

Obstacle Avoidance

Ultrasonic Sensors:

Because the robot will most likely be used outdoors for all practical applications, normal IR cannot be used. IR is inoperable outdoors because of the sunlight. To achieve this problem, ultrasonic sensors were used. Ultrasound sensors work by sending a sound wave out and timing how long it takes for the sound wave to come back. For this project the Devantech SRF04 – Ultrasonic Range finder was used.

Figure 1.Devantech SRF04 Ultrasonic Range Finder

Features:

-Range: 3cm – 3m

-Frequency: 40KHz

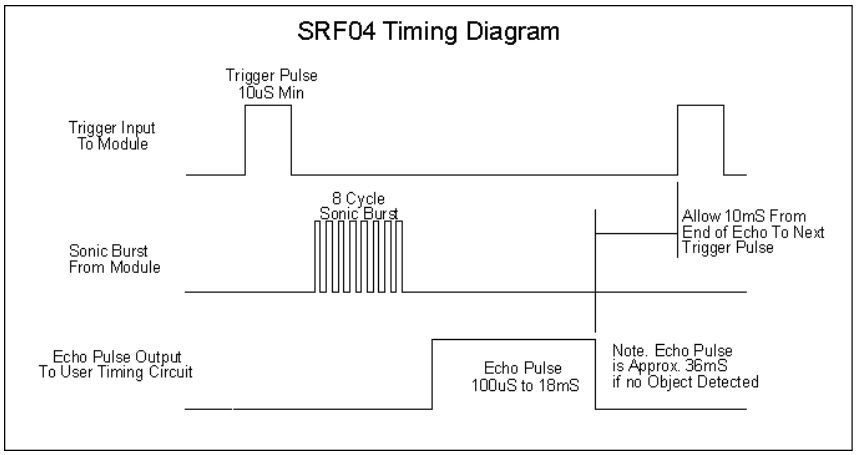


Figure 2.Timing Diagram for Devantech SRF04 Ultrasonic Ranger

Pricing Information

Devantech SRF04 Ultrasonic Range Finder

Cost per sensor: $28.00

Junun Electronics (http://junun.org/MarkIII/)

Bump Sensors

On the outside of the robot, mounted to the physical platform are bump sensors. Bump sensors are basic switches that will be activated when a collision occurs. The bump sensors are tied to I/O pins on the microcontroller that will constantly poll to check if a bump has occurred. Bump sensors are mainly last resort detection of an obstruction in the event that the sonar has failed. I opted to use I/O pins and constantly scan the bump sensors rather than using a voltage divider circuit tied to the analog port because there is an abundance of I/O pins available.

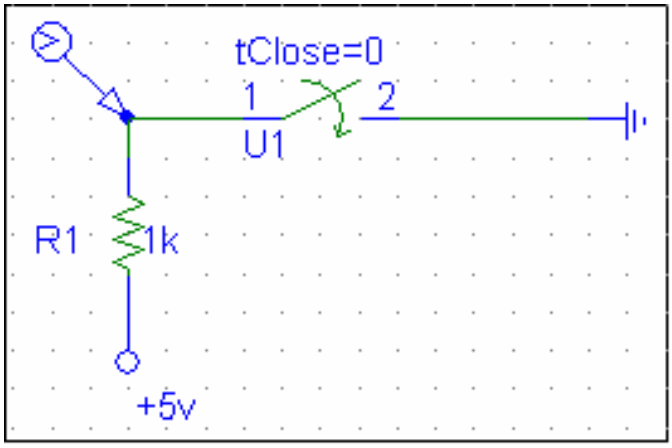


Figure 3.Bump Sensor Circuit

Line Following

Objective- Line following will be used for the robot to find the golf tee. Motors cannot move 'exactly' straight because over time they will become un-calibrated. Line following is used to ensure that the robot will not have to rely on the calibration of the motors. It is also used so that paths other than straight can be used by the robot.

Reflective IR Sensor:

Two reflective IR sensors are used in conjunction with a CdS cell (discussed below) to determine if the robot is on the line. Although mentioned earlier, that IR will not work outdoors, since the sensors are placed beneath the robot the sun will not interfere. The IR sensors work by returning a positive voltage if not detecting a black surface.

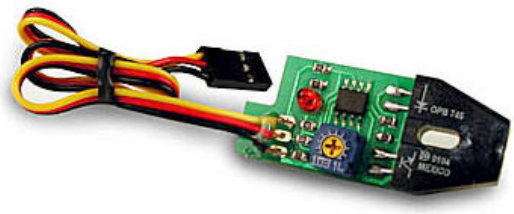


Figure 4.Lynxmotion Single Line Detector

Features

- Range from line: 1/8'' – ½''

Condition	Voltage
Black Line	0V
Other	+5V

Table 1.Reflective IR Conditions

Pricing Information

Lynxmotion Single Line Detector

Cost per sensor: $14.95

Acroname Inc. (http://www.acroname.com)

CdS Cells

CdS cells are variable resistors whose resistance is based on the light that reflects off of it. When used for line following, CdS cells will return a specific resistance based on the light that hits it. To be used in a line following circuit, the CdS cell must be completely isolated and have the only source of light coming directly from the object it is trying to scan. In this case, since it is a line on the ground parallel to the robot and perpendicular to the cell, a shield was made to prevent outside light from affecting it. In order to prevent any other light shining on the line from changing the results, a LED was used to light the area where the CdS cell would be sampling.

Ball Detection

Objective – After placing the ball on the tee, the robot moves back to a safe distance and waits for the golfer to hit the golf ball. The objective of this system must be to accurately detect the release of a ball and notify the robot to deliver another ball. This system must also be dynamic and be able to change to different settings (indoor, outdoors, night, etc.).

RF & CdS Sensor:

The benefit of the CdS cell is the fact that it is able to change based on different conditions. When a golf ball is placed on a plastic tee, a shadow is created over the tee.

When the ball is hit, light is allowed to flood the tee. A CdS cell was placed inside the tee allowing it to give feedback on the condition to the RF transmitter. A voltage divider circuit was then built with the CdS cell and a potentiometer and connected to the RF transmitter. By placing a potentiometer on the circuit, the circuit can be changed based on different lighting systems. Outdoors in full sunlight gave different values from indoors or cloudy lighting.

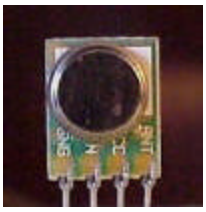


Figure 5.TWS-434A RF Transmitter Module

Figure 6.RWS-434 RF Receiver Module

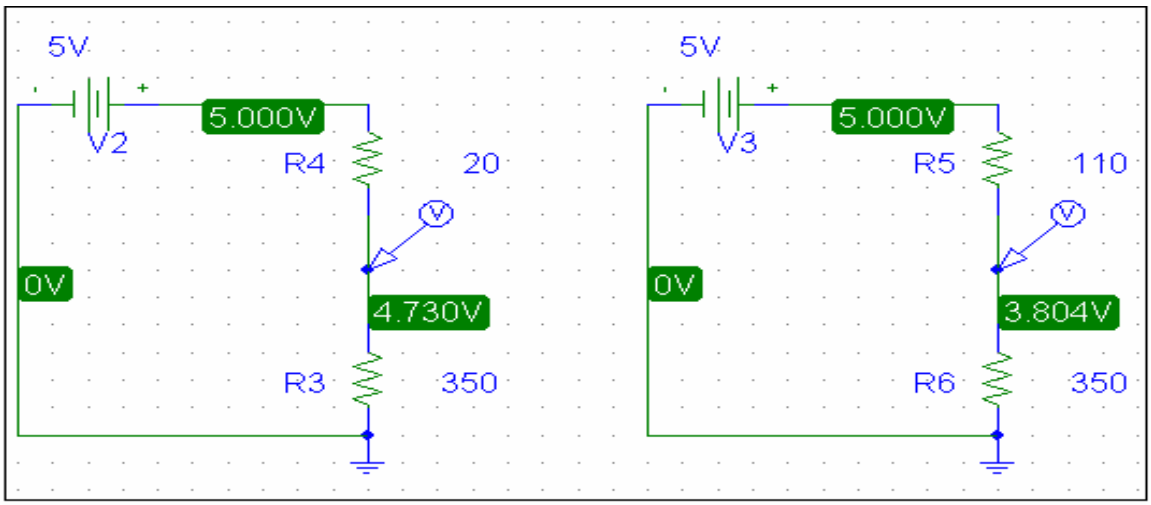


Figure 7.RF Transmitter with CdS Cell Left: Circuit with sunlight Right: Circuit without Sunlight

Pricing Information

TWS-434A RF Transmitter Module

Cost per sensor: $8.95

Reynolds Electronics (http://www.rentron.com)

RWS-434 RF Receiver Module

Cost per sensor: $8.50

Reynolds Electronics (http://www.rentron.com)

Problems and lessons learned- The reason that three reflective IR sensors were not used was because my original idea was to use two and do line following with them. It turns out to be extremely difficult to do line following with two sensors unless the sensors are extremely close together and both very accurate. Thus, near the end of the project I decided to add a CdS sensor between the two IR sensors. Also, always checking power and ground on sensors is a good thing to do. If power and ground is ever reversed, the sensor is basically useless. This occurred with both my sonar modules and required me to purchase two new ones. The biggest lesson I learned is that if I ever went into the chip development industry and wanted to protect a chip from being burned up when the power and ground are flipped, I would spend the extra two cents and add a diode to the circuit. Granted a little power is lost and there is a .7 volt drop, however, it would save chips from being burned up unnecessarily. Though, perhaps this is the industries ploy to get people to buy more chips.

Behaviors

1. Line Finding
2. Line Following
3. Ball Delivery
4. Wait for next delivery

Line Finding

The caddy when initially turned on will drive forward searching for a line to follow. During this period, its collision avoidance sensors act to prevent the robot from hitting anything as well as trying to avoid obstacles in its way. If a sonar detector detects an object in the distance and the other one does not, then the robot will turn away from the obstacle. The two sonar modules are offset by 30 degree's to allow them to pick up a full 180 degrees of range in front of the robot. Unfortunately, with only two sonar devices, if both sonar's sense an object it cannot be distinguished if there is one object in front of the robot or an object on either side of the robot. Therefore, in this case if the sonar's detect an object in both of them, it will try to go straight making the assumption that the object is on the sides. If it turns out that the object is in front (by the object getting closer in both sensors), the robot will then stop and turn around and continue searching for the line. The bump sensors will cause the robot to move in the opposite direction of the activated sensor. Thus if the front sensor is pushed, the robot will immediately back up. Once the line is found, the behavior changes to line following.

Line Following

This behavior assumes that the robot is currently on the line and ready to move towards the tee. Collision avoidance is still used however it is slightly changed in this routine. In the previous routine, the robot was moving randomly to find the line to follow. In this routine, since the robot is already synched with the line, it no longer needs to avoid objects. If it were to compensate its direction to avoid objects, it would immediately be taken of the track. Thus, in this behavior, the bump sensors and ultrasonic sensors cause the robot to stop in its track and wait until the obstruction has moved. Once the robot reaches the end of the line, it will stop and assume it is over the tee. At this point, it transfers to the next behavior of ball delivery.

Ball Delivery

The ball is delivered to the tee in this behavior. Since the robot is now stationary at the tee, the bump sensors and ultrasonic sensors are no longer polled. The ball delivery mechanism is then activated and the ball is delivered to the tee. After the ball is delivered, the robot will follow the line away (Line Following Behavior) from the tee to allow the player to hit the ball. It then stops and wait for the ball to be hit.

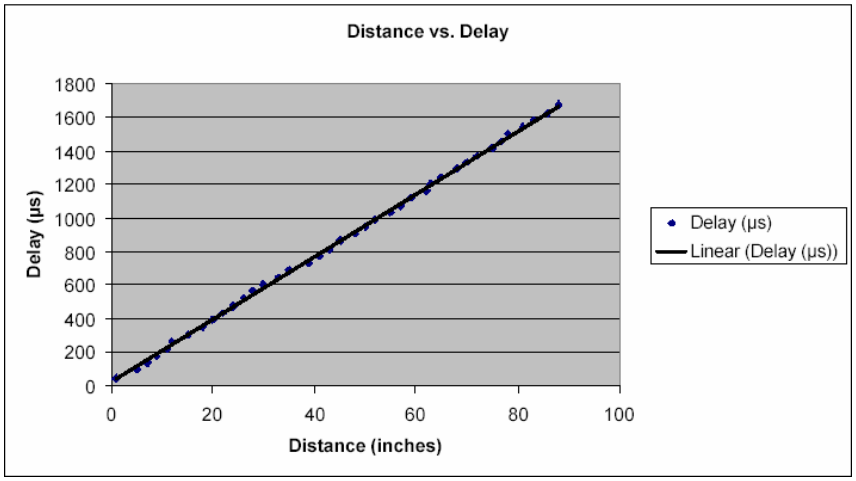
Wait for next delivery

In this behavior, the robot waits for the signal from the RF transmitter telling the robot that the last golf ball had been hit. Once the signal is received, the robot goes forward and follows the line (Line Following) back to the tee and the process is repeated.

Experimental Layout and Results

The first thing tested was the Atmega103 microcontroller. The A/D, I/O, IC, and PWM features of the chip were verified to work.

Next the ultrasound sensors were tested. The ultrasound sensor receives an input from the microcontroller and on the falling edge starts pulsing in order to determine distance. Once the ultrasound signal has been sent, the output pin on the sensor goes high until the pulse is received back. Thus, the Input Capture port on the microcontroller was used to time how long it took for the pulse to return. Shorter pulses correlated to closer objects. The process is very easy to setup however the clock divider on the IC must be manipulated because the internal clock is too fast to pulse the entire length of the sonar sensor. By slowing down the IC clock, the full range of the sonar distance can be realized across a sixteen bit register. Also, both ultrasound sensors had to be synched as they seem to give different values for similar distances.



Data from Steven Theriault: Trebuchet

The bump sensors are pulled high and each are connected to an I/O port that is polled continuously by the microcontroller to determine if it is set. This was the easiest sensor to test.

The RF transceiver circuit was built as described above. The circuit works fairly well to relay the information from the tee to the robot, however because of lighting conditions it tends to fail occasionally as the CdS cell does not trigger a pulse. In order to compensate for this problem, the microcontroller samples numerous amounts of values from the robot in order to ensure that an event has actually occurred.

When the reflective IR sensors were tested, there tended to be many problems with false positives. The IR would return 0V if it 'sees' black and 5V in the absence of black. Unfortunately, when used outside there is a lot of other material that could possibly be on top of the line. In my case, specks of cement littered the line and prevented the sensor from obtaining an accurate reading. The first solution that I obtained for this problem was to sweep the line before using it each time. This proved to work fairly well as most of the false-positives were eliminated. However this is a tedious process that has to be repeated many times during the use of the robot and doesn't seem like it would be practical for a robot that is supposed to be helping you. My final solution to this problem was to have the line following reflective IR to sample the ground 255 times and then determine the surface it was on based on the average result. This proved to virtually eliminate the problem. The only problems that could arise from this method is that the sampling rate and reaction rate of the robot will be severely decreased since so many samples have to be made. However, since the microcontroller runs at 6Mhz and

the fact that this method will eliminate a huge problem with line following, this seems like an extremely worthwhile tradeoff.

Originally, the microcontroller itself was used to drive the servos on the robot. This was achieved by having the motors connected directly to the batter supply and the data pin connected to the microcontroller. A lesson from this was about the destructive nature of motors that are not completely isolated from the microcontroller. When a motor is running at full speed and requires high amounts of current and is suddenly stopped, the current has to travel somewhere in the circuit. For my circuit, that current traveled straight into my microcontroller destroying the internal machinery. After discussing the issue with Uriel Rodriguez and obtaining much needed advice and information, I decided to strip the internal circuitry of the servos and instead use a Texas Instruments SN754410NE chip to control the motors. This turned out to be an amazing improvement in the use of the motors as well as completely eliminating the current issue. For future applications in robotics, these chips would be invaluable as they allow hacked servo's to be used with ease.

Conclusion

I entered this project wanting to build a robot that I could use myself whenever I go to the driving range. After building the robot and testing it, I doubt at this point that the robot would be able to be used outdoors successfully. I was extremely impressed with the ball delivery system as well as the line following system. The ball delivery system worked well to deliver the balls to the same location below it. One of the problems that I could not control was the design of the tees. Because the tees are plastic,

it was very difficult to make the ball stick to the tee. The balls would bounce off the tee extremely easily because of the lack of rigidity. I would change the delivery system in that I would keep the mechanism that gets the ball over the tee, but change the actual dropping the ball onto the tee. I think some kind of gripper system would be much more suitable. Though, if there is anything that made me feel a little better it was the fact that when I tried to put the ball on the tee myself, I myself failed on numerous occasions. My line following system worked well in that the robot followed the line. I think that the system could have been improved tremendously with the addition of CdS cells on the bottom of the robot. The more sensors that are added the quicker deviations could be found and the sooner adjustments could be made. Other line following systems that I found on the internet and from people of past semester that worked extremely well had a minimum of 5 sensors. I would also try to make it so that the sensors didn't have to follow the color black only and could follow any color by calibrating the color to follow when the robot is turned on. This would prove useful if the robot was to be taken to other environments where the color might be faded or other colors were used.

The line finding routine proved to give me many hassles as it was difficult to orient the robot properly so that the line following routine would work flawlessly. I found that if I used bigger compensations to stay on the line in the line following routine it would be counteracted by the robots orientation on the line. If the robot started very close to being straight, it would be thrown off by the line following circuit. The converse was also true in that if the line following circuit allowed for small adjustments, the robot would be off-track if the line finding routine leaves the robot extremely far from the straight position.

I found that moving the RF transmitter between different mediums (home and school), I would get different values. Even though I was able to tune it with the potentiometer on the transmitter, it would seem that at school I was getting negative feedback from an unknown source. This caused problems with detecting if the ball had left or not.

Future Work

For the golf caddy many major improvements would be made if I continued with this project. Sonar sensors would be added to the rear to allow for collision avoidance when the robot went away from the tee. I would also try to improve on the RF transmitter circuit and add a decoder/encoder system to ensure that the transmitted bits were that of the signal and not some added noise. At some point, the eventual goal would be to have the golf caddy to place balls on wooden tees successfully and detect the hitting of balls at a higher percentage.

Acknowledgement

First and foremost I must thank god for without him (or her) I could not accomplish what I have done today. Secondly I would like to remember Tupac Shakur and Christopher Wallace for their influence on my in a very trying time in my life as I got extremely frustrated with this robot. Karl Dockendorf helped me tremendously with much of the code as well as many of the ideas. A lot of the negative criticism that I received was also from him, so I must also thank him for making me want to beat my heads at times. Uriel and Jason played a huge role and helped me debug certain hardware

issues that occurred in the project. Uriel in particular fixed a large problem with my microcontroller and servos. Finally, I would like to throw some shout outs to my homies: Funk Master Flex (Ryan), Special K (Karl), K-Puff (Komal), Fermo, Jules, and Notorious PAL. Holla back!

Program Code

```
#include <io.h>


typedef unsigned char u08;
typedef unsigned long int u16;


u16 adcsample(u08 pinnum)
{
	u16 result;
	outp(pinnum, ADMUX);
	outp(0xC6, ADCSR);
	while(!(ADCSR & (1<<ADIF))) {}
	result = ADCW;
	sbi(ADCSR,ADIF);
	outp(0xC6, ADCSR);
	while(!(ADCSR & (1<<ADIF))) {}
	result = ADCW;
	sbi(ADCSR,ADIF);
	return(result);
}


void delay(u16 delay_time)
```

```
{
 do
  {
    u08 i=0;
    do
      {

      } while(--i);
   } while(--delay_time);
}

u08 checksonar(int sselect)
{
  u08 counter=0;

  cbi(PORTA,0);
  cbi(PORTA,1);

  if (sselect== 0x00)
    {
      sbi(PORTA,0);
    }
```

```
  if (sselect== 0x01)
    {
     sbi(PORTA,1);
    }
  delay(0x0001);
  cbi(PORTA,0);
  cbi(PORTA,1);
  while (bit_is_clear(PINA,4))
    {
    }

  while(bit_is_set(PINA,4))
    {
     delay(0x0001);
     counter++;
    }
  return counter;
}

u08 checkbump(void)
{
  if (bit_is_clear(PIND,2))
```

```
    return 1;
  if (bit_is_clear(PIND,3))
    return 2;
  if (bit_is_clear(PIND,4))
    return 3;
  if (bit_is_clear(PIND,5))
    return 4;
  return 0;
}

u08 checkline(void)
{
  u08 counter,left={0},right={0};

  for (counter=0 ; counter < 255; counter++)
  {
  if (bit_is_clear(PINA,2))
    left++;
  if (bit_is_clear(PINA,3))
    right++;
  }
  if ((left < 150) && (right >= 150))
```

```
    { // Right Sensor On
      return 1;
    }
  if ((right < 150) && (left >= 150))
    { // Left Sensor On
      return 2;
    }
  if ((left < 150) && (right < 150))
    { // At the end of line, Stop!
      return 3;
    }
  // Else, Online, Go straight
  return 0;
}

u08 checkcds(void)
{
  u08 count={0},count2={0};
  u08 res;

  for (count=0 ; count < 0x64 ; count++)
  {
```

```
  res=(u08)(adcsample(0));
  if (res > 0xC0) count2++;
  }
  if (count2 > 0x50) return 1;
  else return 0;
}

void findline(u08 leftfor,u08 rightfor)
{

  u08 fvalue0,fvalue1;
  u08 fchkcds,fchkbump;

  outp(leftfor,OCR0);
  outp(rightfor,OCR2);
  outp(0x00,OCR1B);


  for ( ; ; )
  {

  fchkcds=checkcds();
```

```
if (fchkcds == 1)
{
  return;
}

fchkbump=checkbump();
if (fchkbump == 1) // Rear Bump Sensor Pushed
{
  sbi(PORTD,1);
  cbi(PORTD,0);
  outp(leftfor,OCR0);
  outp(rightfor,OCR2);
  delay(0x5000);
}
if (fchkbump == 2) // Rear Left bumper pushed
{
  sbi(PORTD,1);
  cbi(PORTD,0);
  outp(leftfor,OCR0);
  outp(rightfor-0x20,OCR0);
  delay(0x5000);
}
```

```
if (fchkbump == 3)  // Rear Right bumper Pushed
{
 sbi(PORTD,1);
 cbi(PORTD,0);
 outp(leftfor-0x20,OCR0);
 outp(rightfor,OCR2);
 delay(0x5000);
}
if (fchkbump == 4)  // Front bumper pushed
{
 cbi(PORTD,1);
 sbi(PORTD,0);
 outp(leftfor,OCR0);
 outp(rightfor,OCR2);
 delay(0x5000);
 sbi(PORTD,1);
 cbi(PORTD,0);
}

fvalue0=checksonar(0);  // Right Sonar
fvalue1=checksonar(1);  // Left Sonar
if ((fvalue0 >= 0x30) && (fvalue1 >= 0x30))
```

```
        {
          outp(0x02,PORTD);
          outp(leftfor,OCR0);
          outp(rightfor,OCR2);
        }
        if ((fvalue1 > 0x1A) && (fvalue0 < 0x30) && (fvalue0 > 0x04))
// Something in Right Side
        {
          if (fvalue0 > 0x1A && fvalue0 < 0x21)
          {
            cbi(PORTD,0);
            outp(leftfor-0x15,OCR0);
          }
          if (fvalue0 >= 0x21 && fvalue0 < 0x28)
          {
            cbi(PORTD,0);
            outp(leftfor-0x0F,OCR0);
          }
          if (fvalue0 >= 0x28)
          {
            cbi(PORTD,0);
            outp(leftfor-0x05,OCR0);
```

```
        }
        if (fvalue0 < 0x1A && fvalue0 > 0x13)
        {
          sbi(PORTD,0);
          outp(leftfor-0x15,OCR0);
        }
        if (fvalue0 <= 0x13 && fvalue0 > 0x0C)
        {
          sbi(PORTD,0);
          outp(leftfor-0x0F,OCR0);
        }
        if (fvalue0 <=0x0C)
        {
          sbi(PORTD,0);
          outp(leftfor-0x05,OCR0);
        }
      }

      if ((fvalue0 >= 0x30) && (fvalue1 < 0x04))  // Something
REALLY close to Right Sonar
      {
        sbi(PORTD,0);
```

```
          outp(0x00,OCR2); // Stop right motor
          outp(0x80,OCR0);  // Reverse Left Motor
         }

        if ((fvalue0 > 0x1A) && (fvalue1 < 0x30) && (fvalue1 > 0x04))
// Something in Left Side
        {
          if (fvalue1 > 0x1A && fvalue1 < 0x21)
          {
            sbi(PORTD,1);
            outp(rightfor-0x15,OCR2);
          }
          if (fvalue1 >= 0x21 && fvalue1 < 0x28)
          {
            sbi(PORTD,1);
            outp(rightfor-0x0F,OCR2);
          }
          if (fvalue1 >= 0x28)
          {
            sbi(PORTD,1);
            outp(rightfor-0x05,OCR2);
          }
```

```
  if (fvalue1 < 0x1A && fvalue1 > 0x13)
  {
    cbi(PORTD,1);
    outp(rightfor-0x15,OCR2);
  }
  if (fvalue1 <= 0x13 && fvalue1 > 0x0C)
  {
    cbi(PORTD,1);
    outp(rightfor-0x0F,OCR2);
  }
  if (fvalue1 <=0x0C)
  {
    cbi(PORTD,1);
    outp(rightfor-0x05,OCR2);
  }
 }

 if ((fvalue1 >= 0x30) && (fvalue0 < 0x04))  // Something
REALLY close to Left Sonar
 {
  cbi(PORTD,1);
  outp(0x00,OCR0); // Stop left motor
```

```
    outp(0x40,OCR2);  // Reverse right Motor
   }

   if ((fvalue0 < 0x1A) && (fvalue1 < 0x1A))
   {
    sbi(PORTD,0);
    cbi(PORTD,1);
    outp(0xBB,OCR0);
    outp(0xBB,OCR2);
    delay(0x2000);
    sbi(PORTD,1);
    delay(0x1000);
    cbi(PORTD,0);
    sbi(PORTD,1);
   }
   }
}

void goforward(u08 leftfor,u08 rightfor)
{
  u08 chkline,chkbump,fsense;
  u08 lmotor, rmotor, value0,value1;
```

```
sbi(PORTD,1);
cbi(PORTD,0);
outp(leftfor,OCR0);
outp(rightfor,OCR2);

for ( ; ; )
{
fsense=checkcds();
chkline=checkline();
if (fsense==1)
{
  outp(leftfor,OCR0);
  outp(rightfor,OCR2);
}
if (chkline==1)
{
  outp(leftfor+0x10,OCR0);
  outp(rightfor-0x10,OCR2);
}
if (chkline==2)
{
```

```
  outp(leftfor-0x10,OCR0);
  outp(rightfor+0x10,OCR2);
}
if (chkline==0 && fsense==1)
{
  outp(0x00,OCR0);
  outp(0x00,OCR2);
  return;
}
lmotor=OCR0;
rmotor=OCR2;

value0=checksonar(0);  // Right Sonar
value1=checksonar(1);  // Left Sonar

while ((value0 < 0x04) || (value1 < 0x04))
{
  value0=checksonar(0);
  value1=checksonar(1);
  outp(0x00,OCR0);
  outp(0x00,OCR2);
}
```

```
  outp(lmotor,OCR0);
  outp(rmotor,OCR2);

  chkbump=checkbump();

  lmotor=OCR0;
  rmotor=OCR2;

  while (chkbump != 0)
  {
    outp(0x00,OCR0);
    outp(0x00,OCR2);
    chkbump=checkbump();
  }

  outp(lmotor,OCR0);
  outp(rmotor,OCR0);

  }

}
```

```
void goback(u08 leftfor,u08 rightfor)
{
  u08 chkline,chkbump,fsense;
  u08 lmotor, rmotor, value0,value1;

  cbi(PORTD,1);
  sbi(PORTD,0);
  outp(rightfor,OCR0);
  outp(leftfor,OCR2);
  delay(0x9000);
  delay(0x9000);

  for ( ; ; )
  {
  fsense=checkcds();
  chkline=checkline();
  if (fsense==1)
  {
    outp(leftfor,OCR0);
    outp(rightfor,OCR2);
  }
  if (chkline==1)
```

```
{
  outp(rightfor+0x10,OCR0);
  outp(leftfor-0x10,OCR2);
}
if (chkline==2)
{
  outp(rightfor-0x10,OCR0);
  outp(leftfor+0x10,OCR2);
}
if (chkline==0 && fsense==1)
{
  outp(0x00,OCR0);
  outp(0x00,OCR2);
  return;
}
lmotor=OCR0;
rmotor=OCR2;

value0=checksonar(0);  // Right Sonar
value1=checksonar(1);  // Left Sonar

while ((value0 < 0x04) || (value1 < 0x04))
```

```
{
  value0=checksonar(0);
  value1=checksonar(1);
  outp(0x00,OCR0);
  outp(0x00,OCR2);
}
outp(lmotor,OCR0);
outp(rmotor,OCR2);

chkbump=checkbump();

lmotor=OCR0;
rmotor=OCR2;

while (chkbump != 0)
{
  outp(0x00,OCR0);
  outp(0x00,OCR2);
  chkbump=checkbump();
}

outp(lmotor,OCR0);
```

```
  outp(rmotor,OCR0);


  }


}
void waitforsignal(void)
{
  while(bit_is_clear(PORTD,6))
    {
    }
  return;
}
void dropball(void)
{
  outp(0xCC,OCR1B);
  delay(0x28AA);
  outp(0x00,OCR1B);
}


void initports(void)
{
  outp(0x6C,TCCR0);
```

```
  outp(0x6B,TCCR2);
  outp(0x03, DDRA);
  outp(0xFF, DDRB);
  outp(0x03, DDRD);
  outp(0xA1,TCCR1A);
  outp(0x0B,TCCR1B);
  outp(0x00,OCR1B);
  sbi(PORTD,1);
  cbi(PORTD,0);
  outp(0x00,OCR0);
  outp(0x00,OCR2);

}

int main(void)
{
  initports();
  delay(0x4000);
  delay(0x4000);
  outp(0x99,OCR0);
  outp(0x99,OCR2);
  for( ; ; )
```

```
  {
  findline(0x99,0x99);
  goforward(0x99,0x99);
  dropball();
  goback(0x99,0x99);
  waitforsignal();
  }
}
```