Senior Design Project – iSwiffer 240

Under the guidance of:

Professor Ramesh Karri
Ms. Ellen Daniels

By:

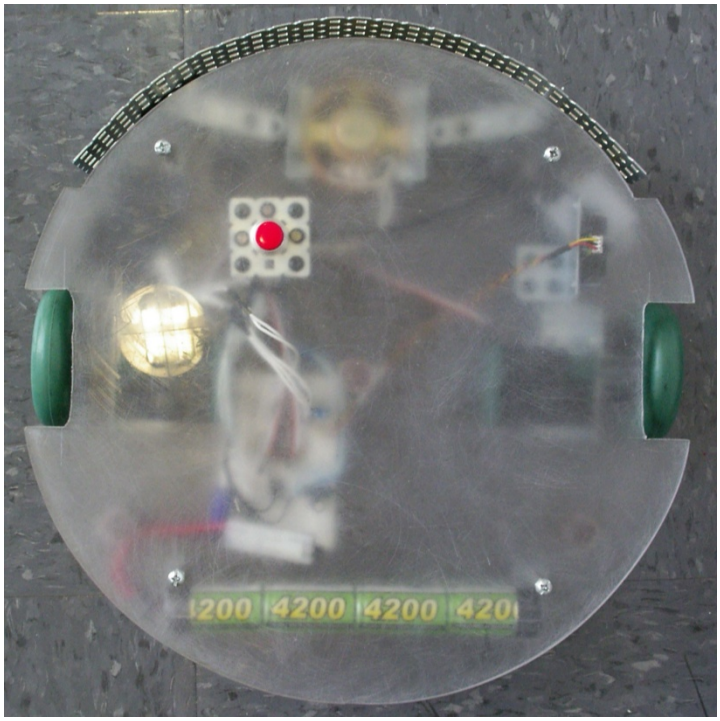
Andrey Ivannikov
Eric Poon
Ilya Brutman

Introduction

This device was built to combat current difficulties of the iRobot Roomba products. The device resembles current features of the current Roomba as their algorithm is pretty efficient at navigating areas of the room.

However, as you know, they use a vacuum cleaner to pick up dirt. This may leave some smaller particles behind. A different approach would be to use an electro-statically charged pad to collect particles large and small off the floor surface. This was accomplished using a product that was readily available, such as a Swiffer pad. A soft rotating brush on the side aids in picking up dirt from small corners and alongside the edges of walls. Swiffer pads are easily disposable and have an astounding capacity for debris.

Besides the Swiffer pad, other product features include things such as overload protection from the high-powered motors, an easily-replaceable battery pack which would allow this robot to run for hours, if not days. A fast charger is also included with this product which would fully recharge a depleted battery in no more than 4 hours.



The device was designed with such high tech features as servo motors for propulsion. This allows the product to select from 256 different speeds per wheel, allowing the highest degree of maneuverability. In order to intelligently follow walls, the product relies on the ability of a modulated infrared signal to detect its distance from barriers of any kind, independent of color, texture, and light pollution.

These, coupled with numerous other innovations make the iSwiffer 240 a truly efficient and user-friendly product, all while keeping costs to a bare minimum. The pages that follow detail the specifics of the iSwiffer 240, complete with progress logs, parts inventory, schematics, et cetera.

Project Log

Inspiration

It started with a few poor, lazy, well-educated college students who would go lengths to avoid the loathsome task of cleaning up. Nobody likes to clean their place, and being Poly students, we were

simply too busy to do it. Having seen the wonders of automation in iRobot's line of products, we decided to see for ourselves how well it works.

Luckily for us, one of our friends has a iRobot Roomba cleaning his room. We borrowed it to see how well it worked and how much of a wonder it really was. Already well-aware of the downfalls of vacuum cleaners to leave dust behind, we were a bit skeptical of its performance. However, we were also quite optimistic, and gave it the benefit of the doubt. After all, it was originally created by our fellow engineering students at the Massachusetts Institute of Technology.

The Roomba had a pretty good cleaning algorithm, able to cover most parts of the room in the amount of time that it would take us to go out and grab lunch. The key word here is *most*. We felt that the algorithm used for cleaning could be improved, and that a lot of areas were covered numerous times, largely due to the shortcomings of using a vacuum brush. Even then, there were particles left behind when the Roomba would claim to have cleaned the room.

Given this, we decided to set out and build our own automated cleaning robot, without the downfalls of a loud obnoxious vacuum cleaner. Another major factor that sets most poor and lazy college students apart from the iRobot Roomba is its prohibitively expensive cost. It would cost over $300 for the base model, and it is just not worth that much money to skip out on cleaning the floor. We said to ourselves that our implementation of the automated cleaning robot will cost under $100 from start to finish. This was later upped to a much more credible $150.

With this, our project came into being: The iSwiffer 240.

Project Goals

- Quick cleaning of area.
- Efficient cleaning of area.
- Intelligent traversal of all areas of room.
- Obstruction-free chassis design so it doesn't get caught when in motion.
- Low-profile design to fit under raised furniture such as coffee tables.
- Good battery life.
- Ease of use; intuitive.
- Reliable.
- Low cost; initial cost, operating cost, maintenance & upkeep costs.
- RoHS compliant – Health and safety is a big concern for us.

Humble Beginnings

When we first started on this project, we made a few simple decisions right off the bat. We decided that the shape of this will be cylindrical, so that it would be able to turn in place without getting caught. We decided that it will be using Swiffer pads to pick up the dirt. We decided that we will be controlling this through the use of a microcontroller. The Swiffer pad and robot shape decisions alone set us off with a good base to work off of. The Swiffer pad measures 10" x 4". Given the provision of a cylindrical

shaped robot, this pretty much determined our circular surface for it to be 12” in diameter, later finalized to be 12.5”.

We decided that it would need a few sensors to navigate around the room. Basing our room traversal algorithm on the Roomba, we decided to implement a touch sensor to detect when we hit walls, and a distance sensor to be able to follow along the edges of the room and furniture. The rest of the navigation is pure dead reckoning with aids from these 2 sensors.

In order to be intuitive and easy to use, we decided to have a simple interface. The iSwiffer240 will only have 1 button: an on/off switch. There will be 3 LEDs and a speaker to let the user know about the status of the robot.

Time for Business – Hardware Decisions Finalized

In the week that followed, numerous choices were considered, and even more decisions were made. We decided to go with a Microchip PIC16F685 to take care of all control and navigation functions. 2 Vex continuous servos were decided upon to actuate the iSwiffer 240. A Lego motor was wired in and custom fitted to drive the side brush. 2 microswitches were attached to the base and activated by a spring-return bumper. A Sharp GP2D120 Infrared Distance Sensor was used to detect distances from the wall. A 5V voltage regulator was first used, which turned out to be highly inefficient and unable to supply the currents required. As such, an OnSemiconductor LM2574N-ADJ adjustable voltage regulator was later configured for 5V and employed to provide constant voltage supply to the microcontroller and the distance sensor. Communication with the user was achieved through a SPST pushbutton on/off switch, 3 bright Red, Green, and Blue LEDs, and a 2” speaker. All this is powered through a 4200mAH 9.6V NiMH rechargeable battery.

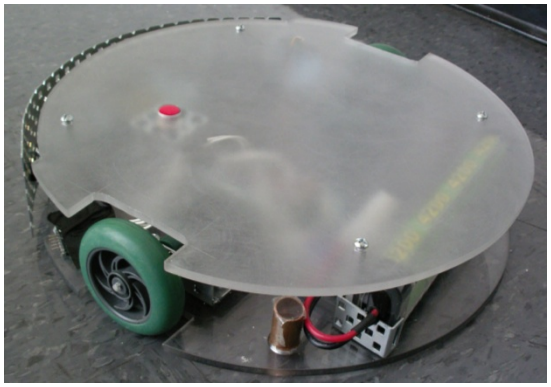
We also designed a nifty mount for the Swiffer pads which enables it to remain in full and even contact with the ground at all times. All this and more will be detailed below.

Buildup Sessions

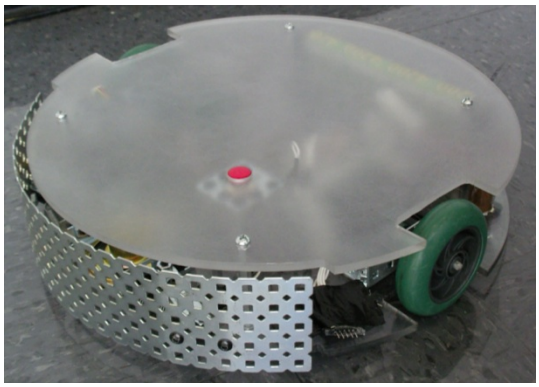
The buildup of the device was no easy task. The challenge that was proposed before us was quite the hefty one, in fact. Within a 12.5” diameter base, we had to fit the circuit board, servo motors, a front bumper, a side brush motor, a distance sensor, and a battery, all while maintaining proper ground clearance and proper weight distribution. Because of our low profile design, the entire robot will be no higher than 3” off the ground, leaving us with only 2” of space inside the chassis to fit all these components. Numerous customized brackets were fabricated to locate all the parts at their optimal locations.

One of the main problems that we had was the bumper design. We always had the concept in mind, but going from concept to production always reveals problems, and this was especially true in this case. The iSwiffer 240 needed to be circular with the button at full extension. To achieve this, we bent the sheet metal for the bumper at the same 12.5” diameter that the chassis uses, and recessed the front section into a 12” diameter arch. The picture of the open chassis below will quickly reveal the cleaver design of an omni-directional front bumper.

Our second main problem with build was wheel mounting. We had chosen to use wheels that were 2.75” in diameter, and while this provided the perfect amount of ground clearance under the chassis, it needed to fit within the 2.5” tall confines of the chassis. To overcome this, we made a cutout for the wheel on the top cover.



The nature of the Swiffer pad called for it to have a light amount of pressure applied to it. We achieved this by having the Swiffer pad dragged along behind it, with the weight of the iSwiffer 240 pressing down on it. The original batteries we had were too light to do this, and it would often tip over onto its heavy front bumper. We overcame this by using a properly weighted battery that would fit pretty far back into the chassis. This will be evident in the picture of the open chassis below.



Our final sizeable problem encountered during the buildup was the user interface. We needed an easily accessible button that would not be obtrusive. We solved this by mounting it just high enough such that it its normal non-depressed state, it would be completely flush with the top panel. The areas around was populated with 3 LEDs which act as indicators of low battery voltage and mode of operation. They were fitted close to the frosted top panel, which creates a nice noticeable display from any direction in the room.

Initial Problems

Like any computer engineer at work, we relied on specifications sheets to fill us in on the behavior of our components. While datasheets are usually correct, there are numerous other implementation factors which affect how closely we get to the specified performances.

The iSwiffer 240, as a robot that drives around consistently each time, needed a reliable way to control the servo motors. Our servos are of the continuous type, utilizing PWM (Pulse-Width Modulation) inputs. The range of input was pulses of 0.5ms to 1.5ms, controlling it to move from full reverse to full forward, respectively. However, the microcontroller that we used would only get the servos to go full reverse at 1ms, and full forward at 2ms. This was obviously not within specifications, and after troubleshooting of all parts, we concluded that the specifications sheets for the servo motors were incorrect. We eventually fixed this problem when we fried the PIC16F685 microcontroller and replaced it with another one. We told it to go full reverse, sending out a pulse of 1ms, and it would just stand still. After some code debugging, we figured to go back to square one and look at the datasheets for the servos. Rewriting our program to pulse 0.5ms to 1.5ms solved the problem. The problem was a defective microcontroller.

With basic driving ironed out, we went ahead to program the procedure which would do most of the cleaning: the outwards spiral. A spiral would be completed by turning the outer wheel at full speed, and then slowly stepping up the inside wheel. But how slowly is slowly? Too fast and it will skip areas between passes, leaving sections uncleaned. Too slow and it will go over certain areas too many times, wasting time. We eventually implemented an exponential speedup of the inner wheel which would allow us to efficiently clean all areas in the spiral. This is described below.

Our original power supply was provided by a generic 5V voltage regulator. This unit would not be able to supply consistent voltage since the servo motors when outputting maximum torque would cause the input voltage to the regulator to drop below what is required to still output 5V. That, coupled with our distance sensor which draws 50mA, would cause the input to the PIC microcontroller to drop suddenly, and effectively resetting itself at somewhat random times. This problem was resolved with the addition of a 470microFarad capacitor in parallel with the power input of the microcontroller to sustain the power for those short bursts. A 22microFarad capacitor was also wired in parallel with the battery to help provide our new voltage regulator with a constant non-fluctuating voltage.

In digital operations, dealing with 1s and 0s is easy, but certain features of our iSwiffer 240 needed to read analog inputs. The PIC16F685 features 10 bits of resolution on its analog input lines, a total of 1024 possible values. After much hunting around on the datasheets for the PIC microcontroller, we finally figured out how to use the feature. We needed the feature for battery level monitoring and the distance sensor. We first started using math to figure out what the proper battery voltage to turn on the Low Battery LED is, but that proved futile, as the values we found, while accurate on paper, was not reflected in the implementation. We quickly discovered that resistors used to divide the voltage of the battery had quite lax tolerances when all used in combination, so we had to figure out the proper 10-bit value through trial and error.

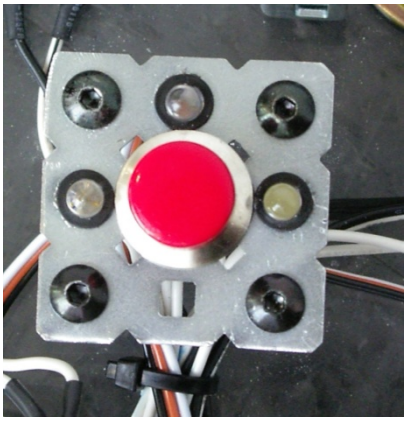
The last problem that we encountered took place on the final day of testing. It seemed like the random reset problem we had in the early days decided to rear its ugly head once again. After much fiddling with the smoothing capacitors, we still had the problem. Incidentally, since it was still being tested, we still had the programming port on it, and it would short the power pin of the programming port to the front bumper on impact, which happens to be ground when depressed, effectively resetting the PIC. After 2 hours of pulling our hair out, we relocated the port for the remainder of testing, and removed it from the finished product. The problem never returned.

Finished Product

After much sweat, blood, and tears, the iSwiffer 240 was finally complete. The testing process showed no weaknesses in our design as of yet. It reliably picks up large amounts of dirt, even after an iRobot Roomba has run through the area. The iSwiffer 240 is easily usable, very straightforward, and most importantly, very rugged. Being the one doing the dirty job of lazy college students, it had to be durable, and durable it was. The chassis is made up of ¼" Lexan sheets, and vertically held up by 4 standoffs. In similar fashion, no other component in the iSwiffer 240 is held on by less than 4 bolts. This concludes our fun times with prototyping this finished product, so feel free to marvel at its over-engineered goodness in the pictures below. ;)

Room Traversal Algorithm & Program

The algorithm for navigating an requiring cleaning was adapted from an iRobot Roomba since it is a very efficient algorithm. The basic modes of it are: spiraling over the starting position, trying to cover the greatest open surface area possible; following a wall or an object it has bumped into while proceding with the spiral; and random bouncing to cover areas in between the spiral and following objects.



Once the iSwiffer is turned on, it goes into a spiral achieved through triggering the motors to run at an exponentially increasing speed, allowing for a spiral with minimal overlap and no missed spot. While in this mode, the green status LED will be lit. As the front bumper is bumped the iSwiffer 240 will proceed into the next mode of following the object it has bumped into at which point the blue status LED will be on. Using a fairly sophisticated algorithm, the cleaning robot will figure what size is the object that it has bumped into and follow with the proper action. If this is a moving object that has moved away from the area where it was bumped, the iSwiffer 240 will briefly clean the small area and move on to the next mode. This algorithm should ensure that the product does not spend an extended period of time cleaning the same spot. The next mode is fairly simple as it traverses the iSwiffer 240 straight until it has bumped into an object, at which point it will turn a randomized number of degrees and repeat while both, the green and blue LEDs are on. The amount of bumps required to exit this mode is also randomized. Once it does exit this mode it tries to find the relative center of the immediate area and proceeds with the spiral. The battery level is tested in between these modes and a

red LED is lit to indicate low power. If battery voltage drops below 7.8V, the robot will refuse to run and turn on the low power level LED.

The programming was done in PICBASIC and the code is as follows.

```
'*****************************************************************
'*  Name    : Senior_Design.BAS                                  *
'*  Author  : Ilya                                               *
'*  Notice  : Copyright (c) 2008 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                                *
'*  Date    : 1/2/2008                                           *
'*  Version : 1.0                                                *
'*****************************************************************
input PORTC.0
DEFINE OSC 4
OSCCON.6 = 1
OSCCON.5 = 1
OSCCON.4 = 0
ANSEL = 0

input PORTC.0

DEFINE ADC_BITS 10
DEFINE ADC_CLOCK 3
DEFINE ADC_SAMPLEUS 50
'=================================================[vars]===========
Speaker     var PORTB.5     'speaker high pin
Bumper      var PORTA.2     'front bumper pin (low when pushed)
DistSensor  Var PORTC.0     'Side distance sensor used for detecting
walls. Analog input
LED         var PORTC.1     'status led pin  high    Green LED
LEDBlue     VAR PORTC.5
LowBattLed  Var PORTC.2     'Low battery indicator high
BlueLed     var PORTB.4     'Blue LED
Mleft       VAR PORTB.7     'left servo pin
Mright      var PORTB.6     'right servo
Brush       var PORTC.4

MiddleOfRoom        var bit         'goes high if the roomba thinks it
is in the middle of the room
ATOD                var byte        'mysterious variable used by
Andrey's function
LOWBAT              var bit         'High when the battery level is
low
SideBumper          VAR BIT         'the side light sensor - change
the name when the light sensor comes in (low when pushed)
NothingInTheYonder  VAR BIT         'Low when no wall close by
i                   var word        'General use loop counter
j                   var word        'General use loop counter
ran1                var word        'used to store a random number
ran2                var word        'used to store a random number
```

```
giveUpCounter       var word        'give up if you're doing something
for to long with no result
timer_ms            var word        'run timer in miliseconds
timer_minutes       var word        'run timer in minutes
incTimerAmnt        var word        'used to pass in values to the
increament timer sub
distance            var word
'=============================================================

'threshold = 29
timer_ms = 0
timer_minutes = 0
PORTB = %00000000
output mleft
output mright
low LED
output led
INPUT Bumper
high ledblue
'input touchsensor

'================================================[notes]=========
'f    VAR   WORD       ' Frequency of note for FREQOUT. bag abb
'A  CON 1760
'B  CON 1975
'C    CON   2093       ' C note
'D    CON   2349       ' D note
'E    CON   2637       ' E note
'F  CON 2794
'G    CON   3136       ' G note
'=============================================================


'turn on status led
high lowbattled
pause 50
high LED
high LowBattLed

freqout speaker, 100, 1980
low speaker
pause 2
freqout speaker, 100, 1765
low speaker
pause 2
freqout speaker, 100, 3141
low speaker
pause 2
freqout speaker, 100, 1765
low speaker
pause 1
freqout speaker, 100, 1980
```

```
low speaker
pause 1
freqout speaker, 100, 1980
low speaker
pause 1
freqout speaker, 100, 1980
low speaker

high brush

middleofroom = 1


low lowbattled
gosub testbat
Home:
    if middleofroom then
        GOSUB spiral
    endif
        gosub testbat
    gosub FOLLOW_WALL
        gosub testbat
    GOSUB RANDOM_BOUNCE
        gosub testbat
    gosub PRE_SPIRAL
        gosub testbat
GOTO Home

END_OF_PROGRAM:
'Send 4KHz tone on Speaker for 100ms
FREQOUT Speaker,100,4000
pause 500
FREQOUT Speaker,50,2000
low brush

low led
low ledblue
low lowbattled
JumpAround:
    sleep 1
    high led
    pause 75
    low led
    high lowbattled
    pause 75
    low lowbattled
    high ledblue
    pause 75
    low ledblue

    sleep 1
    high ledblue
```

```
    pause 75
    low ledblue
    high lowbattled
    pause 75
    low lowbattled
    high led
    pause 75
    low led
goto JumpAround

VERY_LOW_POWER:
low led
low ledblue
high lowbattled

end


FOLLOW_WALL:

        high ledblue
        low led
        'Bumper = 0 when bumper is pressed
        'SideBumper = 0 when sidebumper is pressed

        pause 250
        FREQOUT Speaker,50,2092
        FREQOUT Speaker,50,2348
        FREQOUT Speaker,50,2636
        pause 500

        giveUpCounter = 0

        'follow wall until bumper is pressed 16 times
        for i = 1 to 16
            'drift to the right
            while bumper = 1
            'time to do one loop = ~13ms
            incTimeramnt = 13
            gosub inc_timer

                giveUpCounter = giveUpCounter + 1
                'if i've been spinning around in the circle for too
long, give up.
                if giveUpCounter > 4000 then
                    goto FOLLOW_WALL_EXIT
                endif


                gosub READ_DIST
                if nothingintheyonder = 1 then     'make sharp turn to
the right
```

```
                        low Mleft
                        low Mright
                        pulsout mleft, 155
                        pulsout mright, 100
                        pause 10
                        giveUpCounter = giveUpCounter + 9
                else

                    if sidebumper = 1 then    'drift to right if
SideBumper is not pressed
                            low Mleft
                            low Mright
                            pulsout mleft, 160
                            pulsout mright, 100
                            pause 10
                    else                      'drift to the left if
SideBumper is pressed
                            low Mleft
                            low Mright
                            pulsout mleft, 200
                            pulsout mright, 130
                            pause 10
                    endif
                endif
            wend 'bumper = 1

            giveUpCounter=0

            'at this point we crashed into the wall
            'amount of time to do the rest of the statments = ~340ms
            incTimeramnt = 340
            gosub inc_timer
            'drive backwards a little
            for j = 1 to 10
                low Mleft
                low Mright
                pulsout mleft, 100
                pulsout mright, 200
                pause 10
            next j

            'turn a little to the left
            for j = 1 to 20
                low Mleft
                low Mright
                pulsout mleft, 200
                pulsout mright, 200
                pause 10
            next j
        next i

FOLLOW_WALL_EXIT:
```

```
return


RANDOM_BOUNCE:
    'Bumper = 0 when bumper is pressed

    'bump into walls 5 to 15 times
    high ledblue
    high led
    random ran2
    ran2 = ran2 // 10
    ran2 = ran2 + 5

    for i = 1 to ran2
        'drive full speed forward until front bumper is bumped
        while bumper = 1
        'time to do one loop = ~8ms
            incTimeramnt = 8
            gosub inc_timer
            low Mleft
            low Mright
            pulsout mleft, 200
            pulsout mright, 100
            pause 5
        wend

        FREQOUT Speaker,100,2348

        'time to do the next loop = ~130ms
        incTimeramnt = 130
        gosub inc_timer
        'drive backwards
        for j = 1 to 10
            low Mleft
            low Mright
            pulsout mleft, 100
            pulsout mright, 200
            pause 10
        next j

        'turn a random angle to the left
        random ran1
        ran1 = ran1 // 40
        ran1 = ran1 + 20
        'ran1 is between 20 and 60
        for j = 1 to ran1
        'time to do one loop = ~14ms
            incTimeramnt = 14
            gosub inc_timer
            low Mleft
            low Mright
            pulsout mleft, 200
```

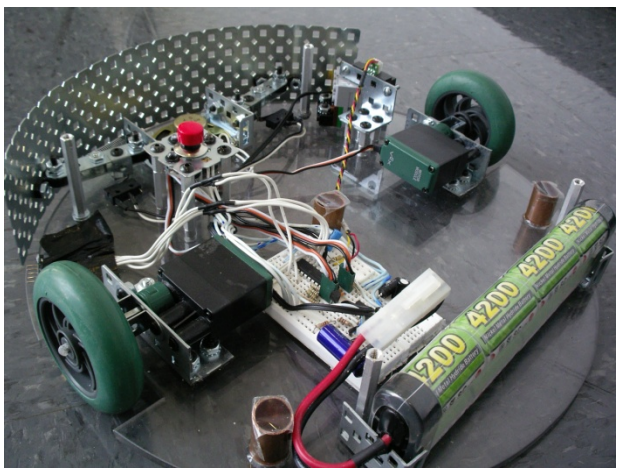
```
            pulsout mright, 200
            pause 10
        next j
    next i
RETURN


SPIRAL:
    '0.5ms is forward
    '1ms is stop
    '1.5ms is backward
    low ledblue
    high led
    FOR i = 1 to 50
        for j = 1 to 8*i
            'time to do one loop = ~13.5ms
            incTimeramnt = 14
            gosub inc_timer
            if Bumper = 0 then   'button is zero when pushed
                goto SPIRAL_EXIT
            endif
            'Send a pulse 0.5mSec long (at 4MHz) to Pin5    [#*10 uS]
            low Mleft
            low Mright
            PULSOUT Mright, 150 - i
            pulsout mleft, 200
            pause 10   'changed to 10 from 40
        next j
    next i
    SPIRAL_EXIT:
return



PRE_SPIRAL:
    ;Runs forward counting duration, until it hits a button
    ;Turns around if duration is big enough, comes back for half
duration
    ;random ran2
    ;ran2 = ran2 // 10
    ;ran2 = ran2 + 5

    FREQOUT Speaker,500,5000
    for i = 1 to 4
        'drive full speed forward until front bumper is bumped
        distance = 0
        while bumper = 1
        'time to do one loop = ~8ms
            incTimeramnt = 8
            gosub inc_timer
            low Mleft
            low Mright
```

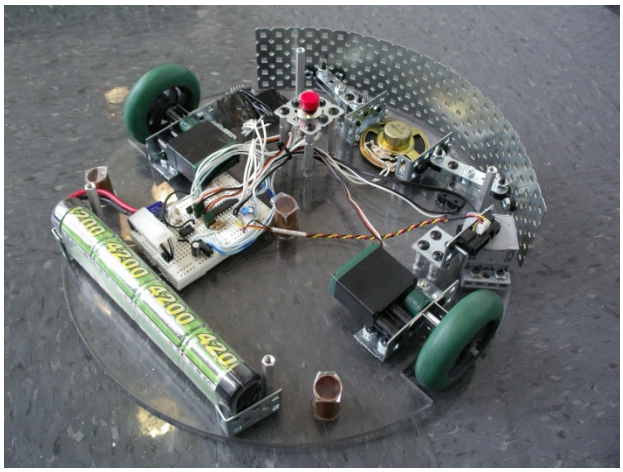
```
            pulsout mleft, 200
            pulsout mright, 100
            pause 5
            distance = distance + 1
        wend

        FREQOUT Speaker,100,2348

        'time to do the next loop = ~130ms
        incTimeramnt = 130
        gosub inc_timer
        'drive backwards
        for j = 1 to 10
            low Mleft
            low Mright
            pulsout mleft, 100
            pulsout mright, 200
            pause 10
        next j

        if distance > 255 then
            'turn around 180 degrees
            for j = 0 to 90
                'time to do one loop = ~14ms
                incTimeramnt = 14
                gosub inc_timer
                low Mleft
                low Mright
                pulsout mleft, 200
                pulsout mright, 200
                pause 10
            next j

            'Drive half the distance forward
            for j = 0 to ( distance / 3 )          'for some reason
its 3 instead of 2. but it works
                'time to do one loop = ~8ms
                incTimeramnt = 8
                gosub inc_timer
                low Mleft
                low Mright
                pulsout mleft, 200
                pulsout mright, 100
                pause 5
                distance = distance + 1
                if bumper = 0 then
                    goto Wall
                endif
            next j

            middleofroom = 1
            Return
```

```
            Wall:
        endif

        'at this point we either hit a wall driving back or didnt find
enough space to spiral
        'turn a random angle to the left
        random ran1
        ran1 = ran1 // 40
        ran1 = ran1 + 20
        'ran1 is between 20 and 60
        for j = 0 to ran1
        'time to do one loop = ~14ms
            incTimeramnt = 14
            gosub inc_timer
            low Mleft
            low Mright
            pulsout mleft, 200
            pulsout mright, 200
            pause 10
        next j
    next i
    middleofroom = 0
RETURN


TESTBAT:
    low PORTC.6
    ADCON0 = %10100101
    INPUT PORTC.7
    ADCON1 = %00010000
    ADCIN 9, ATOD
    ATOD = ADRESL
    IF ADRESH < %10 THEN
        if ATOD < %11100000 then   'if less than 8.4 volts 01010011
         high LowBattLed
         lowbat = 1
        endif

        if ATOD < %10110100 then    ' if less then 7.8 volts
            goto very_low_power
        endif
    ENDIF
    input PORTC.6
RETURN


READ_DIST:
    ADCON0 = %00010001 ' used to be 00011101
    INPUT Distsensor
    ADCON1 = %00010000
    ADCIN 4, ATOD
```

```
    ATOD = ADRESH
                                '20 is a foot away
    if ATOD > 78 then           '75 is an inch away
        SideBumper = 0          'SideBumper is 0 when pushed
    else
        SideBumper = 1
    endif

    if ATOD < 20 then           '(if it's more then about 3 inches
away)
        nothingintheyonder = 1  'nothingintheyonder = 1 when there is
nothing there
    else
        nothingintheyonder = 0
    endif
RETURN

INC_TIMER:
    timer_ms = timer_ms + incTimeramnt
    if timer_ms > 60000 then
        timer_ms = 0
        timer_minutes = timer_minutes + 1
    endif

    if timer_minutes = 3 then
        goto END_OF_PROGRAM
    endif
return
```

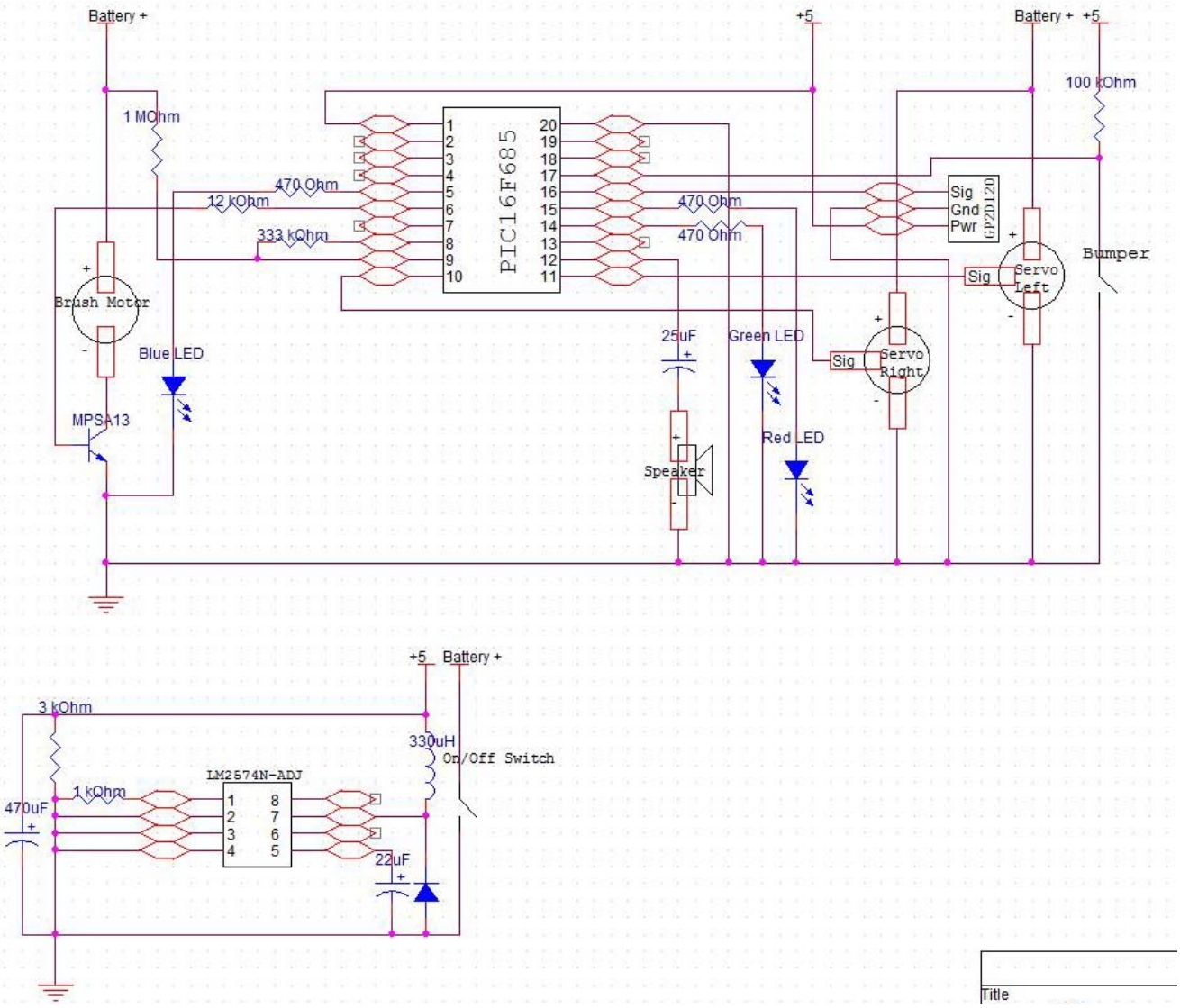
Schematic Diagram

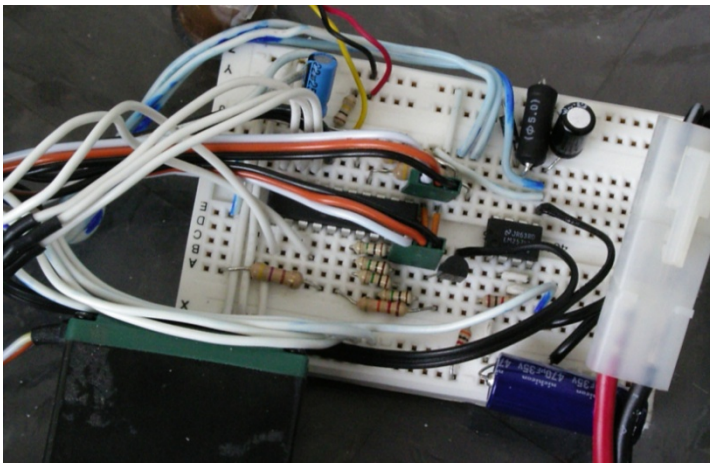




The schematic is constructed of two parts; the first is the voltage regulator circuit and the other is the PIC microcontroller with supporting parts. Most of the components of the voltage regulator circuit were fixed and specified in the datasheet however, some components were changed.

The resistors are used as a voltage divider so that the regulator is adjusted to 5V. The formula for it specifies that the pull-up resistor is between 1kOhm and 5kOhm while the other one needs to be about 3 times larger. The capacitors are used to smooth the input and output voltages and the switch simple cuts/supplies power to the regulator.

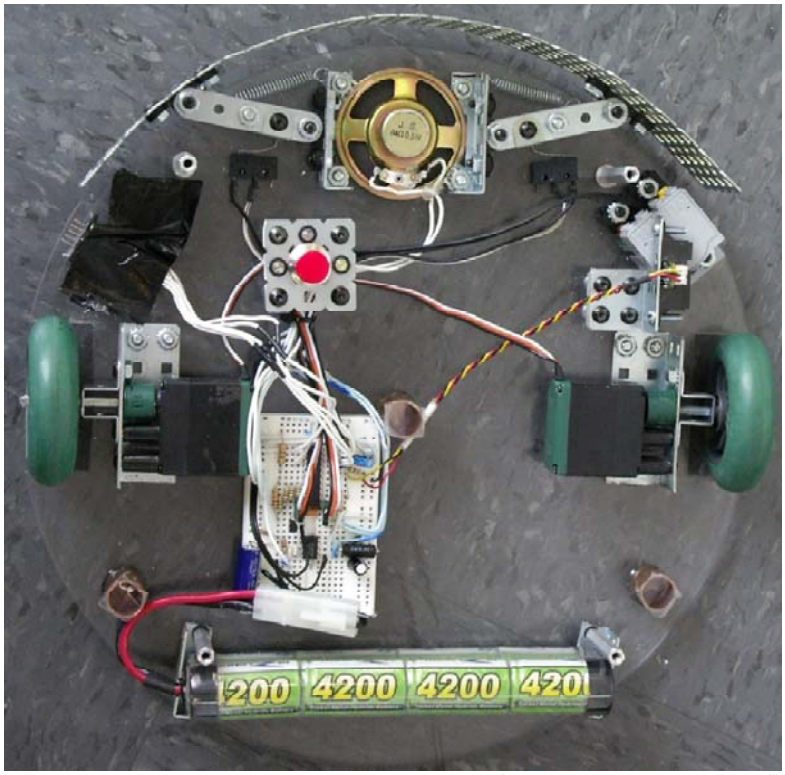




The microcontroller circuit is fairly straight forward. A voltage divider is used to lower the battery voltage so that the microcontroller is able to check it and not burn out. The brush motor is controlled through a Darligton pair transistor, while the speaker is powered through a smoothing capacitor. The bumper button simply shorts pin 17 to

ground when it is pressed while the 100kOhm pull-up resistor ensure 5V on pin 17 otherwise. The rest are just servo motors and the IR sensor with straight forward connections. Since motors require a lot of power, they are directly connected to the battery; the sensor on the other hand requires a constant 5V requiring it to be connected to the voltage regulator.

Chassis Design & Layout



There were few constrictions in how the chassis needed to be laid out. It had to be round and the wheels needed to be at the center for better maneuverability, while the brush motor and distance sensor had to share the NE corner in between the right wheel and the bumper in order to reach corners with the brush and to reliably detect objects. The Swiffer pad was positioned on the south side of the chassis which made it tilt on its wheels. In order to remedy this, the battery pack was placed as far south as possible which also put more of the weight over the pad allowing for more thorough cleaning.

Next, the power button was mounted up front and since most of the wires are run underneath it, the control circuit needed to be at the SW corner which actually helped offset the weight of the brush motor.

The Swiffer pad was suspended using magnets in non-ferric tubes allowing the pad to move vertically while providing for easy removal and installation. Foam was used on the surface that the Swiffer pad mounts to in order to ensure proper contact with the surface.

The bumper located at the front/north of the iSwiffer 240 was mounted on two pivots and held at the neutral position with springs. The pivots allowed it to be pressed from any direction and the very sensitive micro switched are able to detect the slightest bump.

Cost Analysis

The amount spent on the parts used is shown below. The cost of the parts actually used to make the prototype came out to be slightly over $185.

Item	Price	Tax/Shipping	Pkg	Qnt	Total
Distance sensors	$8.99	$10.00	1	1	$18.99
Paint brush	$0.99	$0.08	1	1	$1.07
Misc. nuts and bolts	$3.00	$0.24	1	1	$3.24
Springs	$0.35	$0.00	1	2	$0.70
Vex servo	$19.99	$0.00	1	2	$39.98
Vex wheel	$3.75	$0.00	1	2	$7.50
Battery	$32.25	$8.29	1	1	$40.54
Battery charger	$54.99	$8.99	1	1	$63.98
Voltage regulators	$17.44	$5.95	1	1	$9.44
Grand Total					$185.44

The materials are easily available from such vendors as Digikey and Jameco. It should be fairly easy to locate and purchase them except that some vendors require a customer to place an order for at least 5 or 10 components, which should not be a problem in mass production as the discount provided will be a greater concern since much larger orders will be placed.

Conclusion

Once the iSwiffer 240 was completed it needed to be tested, to do this an iRobot Roomba was allowed to clean an area of floor surface and once it was done, the iSwiffer 240 ran on the same section of floor for 6 minutes. The picture below demonstrates how much dirt the Roomba has missed and the iSwiffer 240 has picked up.

It seems that it is reasonable enough to say that anywhere where the iSwiffer 240 was designed to run, it would outperform a Roomba in terms of cleaning by about 200 percent while running at a pace of about 40 percent quicker, which showcases the reason for the name of the product.

The ability of the product to clean an area combined with the cost of the prototype makes this a successful design and engineering exercise. Future plans for improvement of the product include such items as an easy to access programming port for updating firmware, a charge circuit allowing the iSwiffer 240 to be charge without removal of the battery pack, improved intelligence allowing it to detect an edge on the floor surface as well as the ability to dock to a charging station, and improved protection from dirt with the inclusion of side covering.