ROBOTIC BUTLER: PHASE 1

NSF Summer Undergraduate Fellowships in Sensor Technology (SUNFEST)
Charisma D. Edwards (EE), Clark Atlanta University
Advisors: Dr. Vijay Kumar and Dr. James Ostrowsi

ABSTRACT

The contents of this report describe the specifics of a project aimed to program a “domestic” robot to follow an individual or specified target using sensor technology. Ultimately, the target will be, though not restricted to, the “Smart Chair,” a wheelchair that facilitates physically handicapped citizens with speaking and maneuvering. This would be of significant assistance to those with physical handicaps in the sense that the robot would follow them around to aid in any area possible.

The domestic robot referred to in this project is called the Cye sr. Although the Cye sr comes with its preprogrammed Map-N-Zap software, the decision was made to modify its source code to program the robot directly. The original prototype used one IR receiver and one IR transmitter. After testing, improvements were made involving the addition of only another IR receiver, which required the use of a microcontroller. Upon completion of the project during the allotted period, the goal was generally achieved as the robot essentially follows a target using an infrared sensor. More testing needs to be conducted for further improvements.

1. INTRODUCTION

There are lots of exciting and interesting experiments transpiring in the General Robotics and Active Sensory Perception (GRASP) Laboratory, the robotics laboratory of the School of Engineering at the University of Pennsylvania. Of the numerous robots present, there are a myriad of tests and experiments to be conducted to “upgrade” life and to make it easier. There are several tests already being conducted on “domestic” robots to make these robots more accessible to more people though we cannot do much about the price.

1.1 Goals of the research

The ultimate project goal was to program a “domestic” robot to follow an individual using sensor technology, which, more specifically, means to program a personal robot, the Cye sr, to achieve following behavior within certain constraints. To do this the Cye sr should either move in the direction that the target moves or stop when the target is not detected and when it is too far or too close. The first goal was to make the important and hardest preliminary decisions about the sensor to use as well as the techniques to attempt. After these have been decided, the goal became achieving lemming behavior with the one infrared (IR) receiver. For more accuracy and options, the next goal was to program a microcontroller to interpret the signals of two receivers. The final goal (that time permitted) was to program the Cye sr to achieve following behavior, complete with stopping when needed, using two receivers. Throughout the project considerations had to be taken for a cost restriction of 150 dollars, a weight restriction of 9 pounds and a time frame of 10 weeks with a possibility of project extension.

1.2 Motivation

“Robot following” projects are popular in current technology, but they mostly focus on robots following other robots or groups of robots and forming formations among other things. If robots can be programmed to follow individuals or objects, they could be useful to individuals or groups/companies as carriers for things such as groceries or water or towels depending on the situation or event. The original motivation for this project was in conjunction with the “Smart Chair” project, a project to focused on designing a wheelchair that improve the maneuverability of the physically disabled. With the Cye sr’s capabilities to do tasks, it could follow the Smart Chair and perform various tasks for its operator/user.

2. BACKGROUND

2.1 Cye sr

The Cye and Cye sr are pre-assembled personal robots designed by Probotics, Inc. to perform various tasks for a user. The Cye sr, used in this project, is only an extension of the original Cye robot with more motion techniques via additional sensors. It is approximately 16 inches wide, 10 inches long and 5 inches in height, weighs about 9 pounds (as shown in Figure 1) and moves at 3 feet per second. The Cye requires at least a Pentium 133MHz PC with 19mb of free space, running Windows 95 or 98 but not yet certified for use on the Windows 2000 operating system. It also comes with the capability of interfacing user devices such as on-board cameras, and, for additional costs, wagon and vacuum attachments are available. There is a serial port on the Cye sr that looks like a feed although it exposes the radio link with heavy protocol.

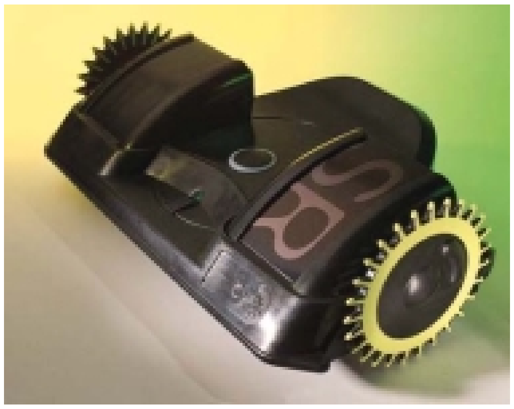


Figure 1. Cye sr

2.2 Map-N-Zap Software

Along with the robot, comes Map-N-Zap software with a user-friendly manual to ensure that the user enjoys the Cye sr with ease. There is no need for a user to be experienced in programming because this accompanying software allows the user to utilize the pre-programmed techniques and commands to operate the robot; however, Probotics, Inc. releases the source code for outside developers to access for programming. The “Map” component of Map-N-Zap provides the interface to control Cye sr. At the same time, it displays a window of the area in which Cye sr is operating and/or navigating. The “Zap” component of Map-N-Zap is a tool provided to create lists of tasks for Cye sr to execute in the form of flow charts. Although there are numerous objects and tools within the software, only a portion of them needs clarification to support the explanation of the methods of experimentation.

In the mapping segment of Map-N-Zap, all five navigation techniques for the Cye Sr and only two of the mapping objects were considered in the preliminary ideas. The two co-navigation techniques are Point-and-Press and Drag-and-Drive while the Point-and-Click, Graph-and-Go and, with the Cye sr, Clap-and-Go are for auto-navigation. With the Point-and Press technique, the Cye can be co-navigated to a point by clicking on that point and holding down the mouse button while with the Drag-and-Drive technique, the Cye is co-navigated by clicking on the Cye sr icon and dragging the Cye along the desired path. When using the Point-and-Click technique, the Cye auto-navigates itself to a point after the user clicks on it. Similarly, the Cye auto-navigates itself to a previously graphed route, called a line path, and proceeds to follow that path after clicking on it. When assigning names to objects on the map, the prompt also allows the assignment for the number of claps, but this option is only available with the Cye sr. If a number of claps is assigned to an object, the Clap-N-Go method may be used. With this method, the Cye sr autonomously navigates to the designated object when it “hears” the number of claps assigned to that object. One of the mapping objects, line paths, has already been introduced. The other mapping object is called a Hot Point; they represent either a particular object in the “real” area or simply a point of reference in the area. These tools were considered to be quite useful to the project had the decision been made to use the software to achieve the goal.

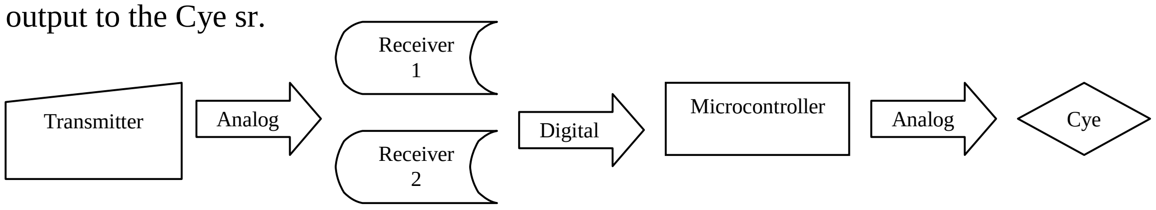
In the zapping segment, there are two commands with several flexible parameters that could be used to “tell” the robot when to move. A command called “Wait…” tells the Cye to wait to perform a specific task or

2.3 Sharp GP2D02 Compact, High Sensitive Distance Measuring Sensor

An important decision in this project was deciding upon the type of sensor to use. The three main sensor types are tactile/touch, machine vision/optical and proximity sensors. The most effective sensor type would be the machine vision/optical sensor; however, as seen in other previous projects, a this type would exceed the most favorable weight restriction. The tactile/touch sensor would work well with the Map-N-Zap’s “nudge” functions, but, as previously stated, it would require too much modification to the source code to use the software; and there still needed to be another sensor added for directional accuracy. Therefore, because of availability, economical cost and disposability, the final sensor system used in this project to make the Cye sr follow the target is the GP2D02, a compact, high sensitive distance measuring infrared (IR) sensor manufactured by Sharp electronics.

2.4 Scenix SX28AC/DP

The SX28AC/DP is a 28-pin microcontroller chip made by Scenix, Inc. (now Ubicom, Inc.) and was programmed in this project to receive the two analog input from the IR receivers, convert the input to digital form, interpret the signals and send a single analog output to the Cye sr.



3. EXPERIMENTATION/METHODS

The initial idea was use the Map-N-Zap software and its provisions to program the Cye sr to follow the given target, so the first task was to become familiar with the software and how it works with the Cye sr. Using the techniques and tools mentioned above, there were a number of methods of experimentation considered. The zap files discussed previously would be of optimal use in the following preliminary ideas.

One method would be to map the specified target as a Hot Point. If the movement of the target could be, in some way, programmed to be mapped on the software as it moves, the Cye could be directed to move to that Hot Point continually. Another method could be to use a sensor to “track” the movement of the target as a line path.

In a zap file, the Cye could be directed, using any of the co-navigation techniques, to traverse through that path continually. The hardware also provides freedom to use outside input, which can be used within the Map-N-Zap program. The zap files have commands that instruct the Cye to act depending on specified inputs. The Cye could be instructed to wait until nudged or it receives a high or low input to perform a specific task, which could be as simple as making a specified sound. Although the software provides the Cye sr with many useful capabilities, it had a number of limitations as well. If the software were to be used, much of the source code for the Map-N-Zap software would have to be modified. The extents of these modifications were unknown because of the unknown dependencies that some files had on others. For this reason, the decision was made to program the robot directly. Because of previous projects with the Cye robots, certain useful functions had already been extracted from the original source code and collected in a new source program.

After modifying this program to receive analog input and turn on the output, hardware modifications were made on the Cye sr to receive input analog input from an infrared receiver. After testing, there needed to be another receiver added for more accuracy. Since there would be only one transmitter and two receivers, there needed to be a microcontroller to distinguish between the two signals before the Cye sr reads them. Hence, the microcontroller, discussed previously, was programmed to receive the two outputs from the IR receivers (referred to as microcontroller input). Both microcontroller input voltages were divided in half, and one voltage was offset enough to be distinguished from the other signal. The microcontroller's output toggled between its two adjusted input values at a rate slightly faster to the Cye sr's reading speed (a matter of milliseconds), so it practically received the updated status of both receivers simultaneously. A resistor and capacitor were also needed to convert this output signal to the Cye sr back to analog data.

After this extensive testing and re-programming was completed, more hardware alterations were made; the two receivers were mounted on the Cye sr and connected to the power supply just as the other with the single receiver. The source code then needed to be adjusted to instruct the Cye sr to move according to the new microcontroller output data.

4. RESULTS

The initial attempts to use the Map-N-Zap software to program the Cye seemed easier in the sense that the software was already designed to receive outside input data and act accordingly. Because of some provisions and restrictions and the lack of certain options, the source code would have to be adjusted to include the options needed. If this were done, adjustments would also have to be made to other programs in the software such as graphics programs that may be affected by these changes. After modifying the previously compiled direct communication (with Cye sr) program and adapting the hardware, tests were run to observe sample data in different directions. Because there was only one receiver initially, the IR receiver was only going to read low or high, implying that the IR transmitter was detected or not, respectively (i.e. the IR receiver is active low). In this case, the IR receiver was mounted facing one direction allowing the signal to only be low when the IR transmitter is detected in that direction. Otherwise, the transmitter would not be detected, causing the signal to be high. The Cye sr was then programmed to turn and move in the direction of detection, or, otherwise, to turn and move forward in the opposite direction. Testing and improvements were made to make this lemming behavior more accurate at as great a distance as possible. Different movement functions were attempted for optimal behavior. Although there were already functions to tell the Cye to turn and move or move in a specified direction, it was best to control the speed of the wheels, which also had to be tested. For greater detection distance, the frequency of the IR transmitter needed to be as near to 40 Hz as possible. With this sensitivity, signals in the opposite direction had to be blocked to avoid inaccurate signals. (Note: These ranges were of a spherical form as opposed to linear. If we recall that in measuring spherical coordinates, ρ is the distance to a point, θ corresponds to the latitudinal polar angle and ϕ refers to the longitudinal angle.) The angles θ and ϕ of the IR receiver also affected the range of detection. After testing for optimum speed, accuracy and distance, the Cye sr was able to follow the “target” (the IR transmitter mounting) quite accurately within more than seven feet (ρ) for every angle of θ and all positive angles of ϕ. Although this achievement was at such a great distance and with such accuracy, the signal was only detected in one direction. Therefore, the Cye sr was instructed to do the same whether it was receiving a signal in the opposite direction and when it was not receiving a signal at all (i.e. the IR transmitter was turned off/not powered). After much testing and correction, the microcontroller program ran successfully with simulated, fixed and actual data. The prototype program was created to instruct the Cye sr to move accordingly if the signal was detected from either direction (right or left) and to stop if neither receiver was receiving a signal.

5. CONCLUSIONS

The robot following project using a “domestic” Cye sr robot was achieved by communicating with the robot directly (instead of using the accompanying software) and improved upon successfully. Because of time constraints, testing for optimum distance, precision and accuracy were not performed but are in preparation. Similar tests as with the initial prototype can be duplicated to achieve this targeted behavior.

6. RECOMMENDATIONS

Because the project is geared toward continuation, the project could be continued in a number of possible ways. More testing should be done to achieve the accurate and consistent following behavior over larger distances. There also needs to be provisions made for the Cye sr to stop when it has come too close to the target. This may be done by shielding the IR transmitter from the receivers as the robot comes directly beneath or at a certain angle underneath the transmitter. Once the ideal following behavior has been achieved, this may be attempted with numerous robots at the same time to use as carriers for individuals or group functions. The same project could also be attempted with or used within the Map-N-Zap software to use in the Smart Chair project. In this case, the Cye sr would follow the Smart Chair (with the IR transmitter mounted in an most advantageous position), and the operator would be able to send it to do perform tasks using the Map-N-Zap software.

7. ACKNOWLEDGEMENTS

First, thanks goes to the National Science Foundation for funding the Summer Undergraduate Fellowship in Sensor Technologies (SUNFEST) program, and thank the SUNFEST faculty and staff for the opportunity to participate in this research and for all the help, support and guidance. Thanks for all of the time and efforts go to Professor Nathan Ensmenger for the ethics seminars and to Dr. Dwight L. Jaggard for the presentation training and to Janice Fisher for writing guidance. Great appreciation goes to the University of Pennsylvania for supplying all of its facilities throughout the entire summer.

Last, but certainly not least, a big thanks goes to the entire GRASP Laboratory faculty, staff and team. Special thanks go to Greg Grudic, Tetsuya Hasebe and Mark Berman for all the assistance. Many, many thanks go to Allen Miu for giving the sensor research direction that no one else seemed to be able to do. Countless thanks go to Daniel Walker for everything. Finally, to advisors, Dr. Vijay Kumar and Dr. James Ostrowski, thanks for the encouragement, support, guidance, teaching, checking, etcetera.

8. REFERENCES

[1] Everett, H. R, *Sensors for mobile robots: theory and application*. Wellesley, Mass.: A.K. Peters,1995.

[2] Massachusetts Institute of Technology, Embodied Intelligence (6.836), CENTIPEDE Project. Information available electronically at http://web.mit.edu/eishih/www/courses/6.836/, 2000.

[3] Sharp Electronics Corporation. *Sharp GP2D02 Compact, High Sensitive Distance Measuring Sensor.* Technical Documentation, 2001

[4] Ubicom, Inc. SX Family User's Manual, Revision 2.0. Information available electronically at http://www.ubicom.com/pdf_files/techdocs/sx_usermanual.pdf, 1999.

APPENDICES

9.1 Appendix A

```
                device SX28L,OSCXTMAX,TURBO,STACKX_OPTIONX,SYNC

        reset   start           ;goto 'start' on reset

                org     8

outerloopctr    ds      1
pwmctr          ds      2

                org     $10

ontime1         ds      1
ontime2         ds      1
offtime1        ds      1
offtime2        ds      1
inputctr        ds      1
w1on            ds      1
w2on            ds      1
w1off           ds      1
w2off           ds      1
temp            ds      1
deltemp         ds      1


                org     0

;this interrupt records the number of times
;that the input pins are high and low (and saves it)

                bank    $0

                snb     rc.1
                inc     ontime1         ;
                snb     rc.2
                inc     ontime2         ;
                sb      rc.1
                inc     offtime1
                sb      rc.2
                inc     offtime2
                decsz   inputctr
                reti
                mov     inputctr,#120
                mov     w1on,ontime1
                mov     w2on,ontime2
                mov     w1off,offtime1
                mov     w2off,offtime2
                clr     ontime1
                clr     ontime2
                clr     offtime1
                clr     offtime2
                reti


start
                mov     w,#%10001000
                mov     !OPTION,w
                mov     inputctr,#200

                bank    0
                mov     !rc,#%00000110      ;prepares all pins of port c to be output
                mov     !rb,#0

:loop
                ;mov    pwmctr+1,#1
                jmp     :pwm1outer

:pwm1outer
:pwm1           setb    rc.0            ;toggles all pins in port c
                mov     temp,w1on
;               add     temp,#15        ;need to take out this comment
                clc
                rr      temp
                mov     w,temp          ;moves value into working register for delay
                call    delay           ;calls delay subroutine
                clrb    rc.0            ;toggles all pins in port c
                add     temp,w1off
                add     temp,#20        ;need to increase number(s)
                mov     w,temp
                call    delay
                decsz   pwmctr
                jmp     :pwm1
                decsz   pwmctr+1
                jmp     :pwm1outer
                ;mov    pwmctr+1,#1
                jmp     :pwm2

:pwm2outer
:pwm2           setb    rc.0            ;toggles all pins in port c
```

```
                mov     temp,w2off
                clc
                rr      temp
                mov     w,w2on
                add     w,temp
                call    delay           ;calls delay subroutine
                clrb    rc.0            ;toggles all pins in port c
                mov     w,temp
                call    delay
                decsz   pwmctr
                jmp     :pwm2
                decsz   pwmctr+1
                jmp     :pwm2outer
                jmp     :loop

delay
                mov     deltemp,w
                test    deltemp
                snz
                ret
                mov     outerloopctr,w
:outerloop      decsz   outerloopctr
                jmp     :outerloop
                ret
```

9.2 Appendix B

```
//practice.cpp : Final Visual C++ program used in Robotic Butler: Phase I project.
//Charisma D. Edwards, SUNFEST 2001 and Daniel Walker
//Advisor(s): Dr. V. Kumar & Dr. J. Ostrowski
//Other Acknowledgements: (Mark Berman,) Greg Grudic, Tetsuya Hasebe

#define CYESRV1_EXPORTS
#include "stdafx.h"
#include "cyesrv1.h"
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <iostream.h>



int main(int argc, char* argv[])
{
//Creates an output file to monitor Cye's actions.
        FILE *logfile;
        int analog, left, right, temp;
        float pwm1, pwm2, lastpwm1, lastpwm2, goodpwm1, goodpwm2, ratio;
        logfile = fopen("Testing.txt","w+");
        printf("hit enter to create a cyesrv\n");
        getchar();
        CCyesrv1 *fred = new CCyesrv1(19200, 1);
        fred->ClearObstacle();

//Assures that Cye remains stationary and resets it's starting position.
        fred->SendMotorVelocities(0, 0);


//Tells Cye to move as long it receives input .
        fred->SendRequestAnalog();
        analog=fred->GetAnalog();

        while(!_kbhit()){
        //fred->SendOutputOn(TRUE);
                fred->SendRequestAnalog();
                analog=fred->GetAnalog();
                temp=analog&0xffff;
```

```
                    if(temp < 450)
                        pwm1 = temp;
                    else
                        pwm2 = temp;

                    //if(abs(pwm1 - lastpwm1) < 0.05)
                        goodpwm1 = (int)(pwm1/405);
                    //if(abs(pwm2 - lastpwm2) < 0.05)
                        goodpwm2 = (int)(pwm2/289);

                    //lastpwm1 = pwm1;
                    //lastpwm2 = pwm2;

                    ratio = (goodpwm1/goodpwm2)*10;

cprintf("\nLeft %i\tRight %i\tRatio %1.1f\t",(int)goodpwm1,(int)goodpwm2,ratio);


                    switch((int)ratio){
                        case 5:
                            right=0;
                            left=0;
                            cprintf("Stop\n");
                            break;
                        case 10:
                            right=25;
                            left=100;
                            cprintf("Right\n");
                            break;
                        case 0:
                            if((int)goodpwm2 == 2){
                                right=100;
                                left=25;
                                cprintf("Left\n");}
                            else{
                                right=100;
                                left=100;
                                cprintf("Straight\n");}
                            break;
                        default:
                            break;
                    }
                    fred->SendMotorVelocities(right, left);

                }
```

```
        fred->SendStopMotors();
        cprintf("world ended\n");
        return 0;
}
```