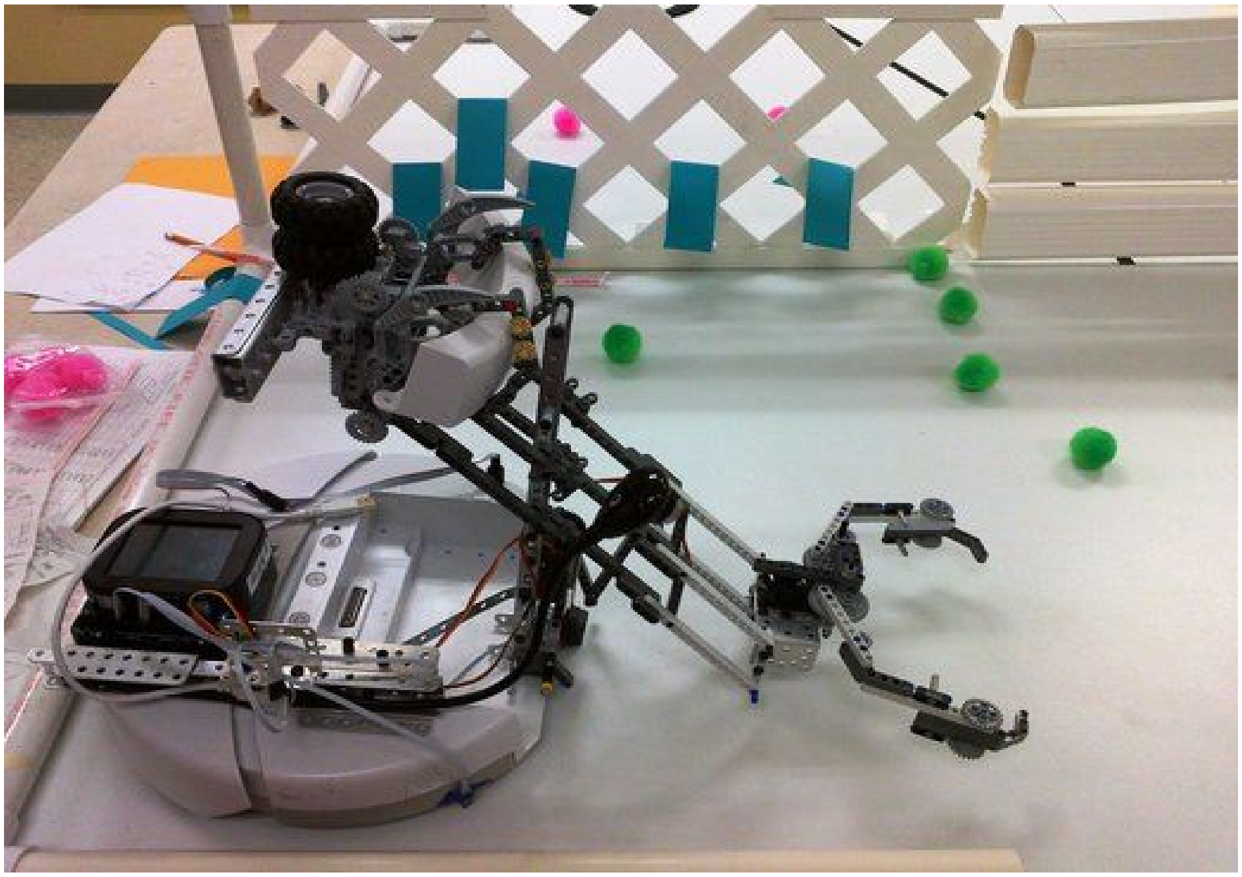
PIPE: Programming in Pure English

The Conversion of English-language Sentences into Compilable Code

Question

Is it possible to simplify the programming of robots by designing an application that can accurately translate English-language sentences into compilable code in distanced traveled, and



command accuracy?

Hypothesis

If an application that can translate English language instructions into compilable code that a robot could download and execute, then the robot will be able to follow then translated instructions accurately.

Background Information

Machine code is a language directly understandable by a computer's central processing unit, and all programs must be in before they can be run. After the source code for a program has been written by humans in a programming language, it is compiled into machine code through a compiler. At an early age, many American children learn the English-language. Later, children begin learning programming languages such as C in middle school or high school. However, although it is a concept discussed in college, machine code is difficult for humans to read and produce because it consists of binary. Thus, programmers use programming languages. In this project, the English-language will be used to write programming code, which will be compiled into machine code for robots.

Materials

-KISS IDE

-Compiling software that uses C such as Code::Blockers

-iCreate

-iCreate USB Cable

-Measuring Tape

-Laptop

-USB Flashdrive

Abstract

There are two aspects to a robot-- the electromechanical machine (or "body") and the computer ("brain"). While some people are interested in designing and building robots, others are more expert at or passionate about programming the logic that will guide and control a robot. Programming robots can be slow and challenging, especially to beginners in the field of robotics or just everyday
people. If the creation of an English-language code translating application is successful, then it should be of great assistance to those struggling with existing graphic and text-based programming languages, in various settings, whether it be in a classroom or an office. In essence, the purpose of this project is to create a application that can translate English-language sentences into code.

Procedures

1) Install KISS IDE and a C compiler on a computer.

2) Familiarize and prioritize movement functions for the iCreate to be translated as well as basic programming concepts in C.

3) Translate the prioritized functions into an English-language
equivalent based on the desired command and highlight keywords from the English-language equivalent.

4) Open the C compiler and initialize recognition strings for the translations based on the declared keywords.

5) Use FILE *file_pointer for the translation to be saved in .txt.

6) Use fprintf (file_pointer, "int main(){\n create_connect();\n"); to ensure int main() and create_connect() will be in the .txt file.

7) Use char word[12]; char read = ' '; int i, a; to initialize variables.

8) Code the desired functions to be translated through the use of the recognition strings declared if inputted.

Ex. If you said “char forward [7] = “forward”;” then use the following:

```
if(read == 'f'){

i = 1; word[0] = 'f';

while(read != '\n' && read != '.' && read != ' ' && read != ','){

        read = getchar();

        word[i] = read;

        i++;}
a=0; i=0;

while(i<7){

        if(word[i]!=forward[i])

    a++;

    i++;}

if(!a){

read = getchar();

while(read != '0')

read = getchar();

scanf(" %d", &yes);

fprintf (file_pointer, "icreate_forward(%d);\n msleep(500);\n", yes);          }

}
```

9) Proofread the program any errors (missing semicolons, missing brackets, etc.) and correct them.

10) Repeat step 8 and 9 for every prioritized function.

11) Add commands that will close the application and close ensure that the file pointer closes after the application closes.

Ex.

```
if(read == 'z'){

     fprintf(file_pointer, "}\n", yes);

     fclose(file_pointer);
```

```
        return 0;

        break;

    }

}

fprintf(file_pointer, "}\n", yes);

fclose(file_pointer);

return 0;
```

12) Compile and program and run it; if the program does not compile or run correctly, repeat steps 8, 9, and 10. Ensure that the program creates a .txt file with the translated commands.

13) Test the program for the distance traveled for movement functions with forward and back being two separate tests.

14) Run the test until a total of twenty trials have been completed per function.

15) Analyze if the robot did correctly travel to the desired distance by measuring the length of displacement.

References

Bergin, J. (2008, October 11). Gui Programming in Java v2.
Retrieved January 10, 2014, from
http://csis.pace.edu/~bergin/sol/java/gui/JavaGUI.html#RTFToC2

Code.org (2013, February 26). What Most Schools Don't Teach
[Video file]. Retrieved from!!http://www.youtube.com/watch?v=nKIu9yen5nc

Gates, B. (2012, July 11, 2012). Improving Education in the U.S.
Retrieved December 27, 2012 from
http://www.thegatesnotes.com/GatesNotesV2/Topics/Education/
Improving-Education-in-the-US

Hutchins, J. (1993). Language Translation. Encyclopedia of
Computer Science, 733-738. Retrieved from
http://www.hutchinsweb.me.uk/EncComputerScience-1993.pdf

Lazerowitz, R. (2013, May 12). What is Natural Language
Processing. Retrieved December 20, 2013, from
http://infospace.ischool.syr.edu/2012/05/11/what-isnatural-
language-processing/

Liu, H. (2004). English: The Lightest Weight Programming
Language of them all. Retrieved on December 30, 2013, from
http://web.media.mit.edu/~hugo/publications/papers/LL4-hugoe
nglish-lightestweight.pdf

Data

Accuracy of PIPE's icreate_forward

As the predicted distanced traveled increases, the actual distance travelled decreases. Some factors that lead to this was the friction the wheels and floors. The iCreate only reads meters, so the conversion of centimeters to meters in this project without the use of another program. 1 centimeter is .01 meter. As the constant is multiplied, more error within the actual distanced traveled occurs.

Data Analysis

By creating PIPE, I have shown that it is possible to create an application that allows users to translate English language sentences to code accurately under a set of rules and conditions. PIPE will only be able to translate movement commands in this stage of the framework. Users need to have a basic understanding of turns and angles to utilize the "turn", "left", and "right" functions correctly and users must also be able to construct English-language sentences. Users of PIPE will also have to know that all number values for the forward and backward commands are recorded in meters despite what units the user specified. Lastly, within PIPE, users must enter "z" and press the ENTER key to end the application and number values must always begin with an 0.

Framework

I had not worked in this field of computer translation before, so I had to start with some research in Natural Language Processing (NLP), a field of computer science and linguistics concerned with the interactions between computers and the human language. To ensure that PIPE is able to translate English-language commands, I used “forward”, “back”, “left”, “right”, and “turn” for my character strings and allowed only tested for movement. Originally, the application’s goal was to be able to translate English to C and compile right onto the KIPR of the iCreate, but after research, evidence proved that a “direct compiling” is impossible because the KIPR does not have a built in compiler. With this, the idea of saving the translation onto a .txt file which a user could save as a .c file and manually copy onto the KIPR.

Conclusion

From the data, it is proven that an application that can translate English language instructions into compilable code for a robot to download and execute. Through the use of string declaration and recognition commands, an application can be designed to recognize and translate English-language sentences into compilable code. Although only movement commands were included in the application, it is possible to translate most of the iCreate’s known commands accurately despite errors in distance caused by the wheel of the iCreate. In addition, through further investigation and the inclusion of more definitions, it is found that the translation logic can perform a multitude of tasks, especially utilizing sensors and other hardware. In essence, with preset rules such as starting number values with 0 and using ‘z’ to end the program, it is possible for English-language commands used by humans to be translated into compilable code for a robot to read in machine language.

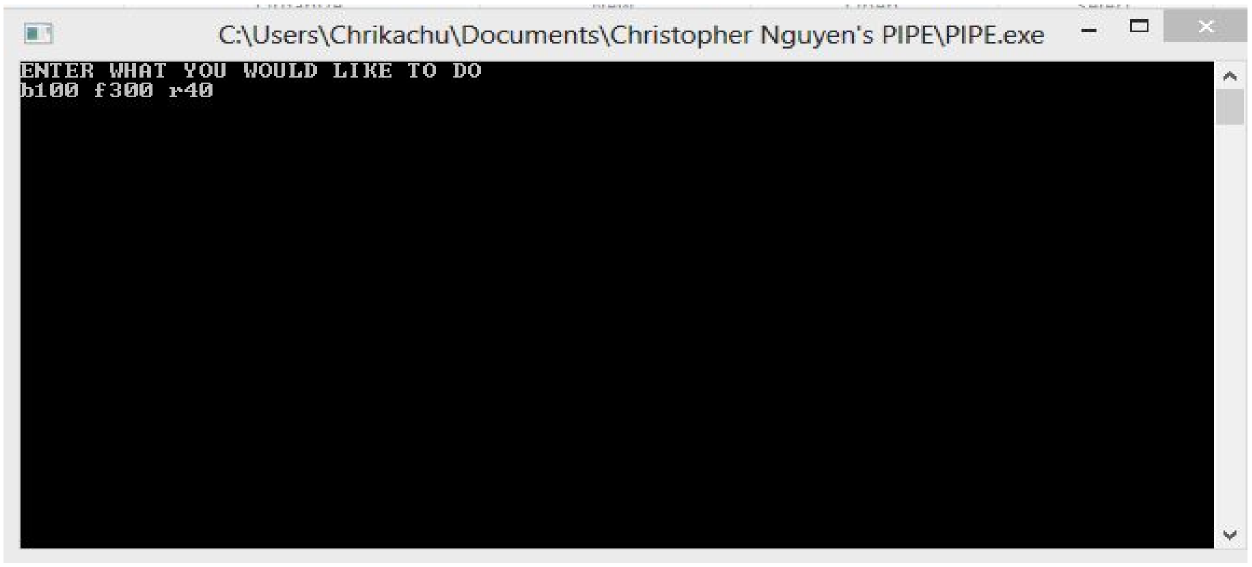
Future Steps

With an application that can translate English into code that a robot can read in machine language, more possibilities open up. The iCreate does not only possess movement functions, but it also has other hardware such as a line sensor, touch sensor, and .mp3 player. Legos and VEX robotics parts can also be attached to the iCreate to perform additional functions through by programming additional motors and sensors. With the knowledge of this application's success, more commands can be translated. The iCreate is an introductory platform to robotics for many. Testing of a more refined version of PIPE should encourage the development of smart-compilers that will not only translate English into code, but also English to machine language for everyday users.

First Preliminary Test

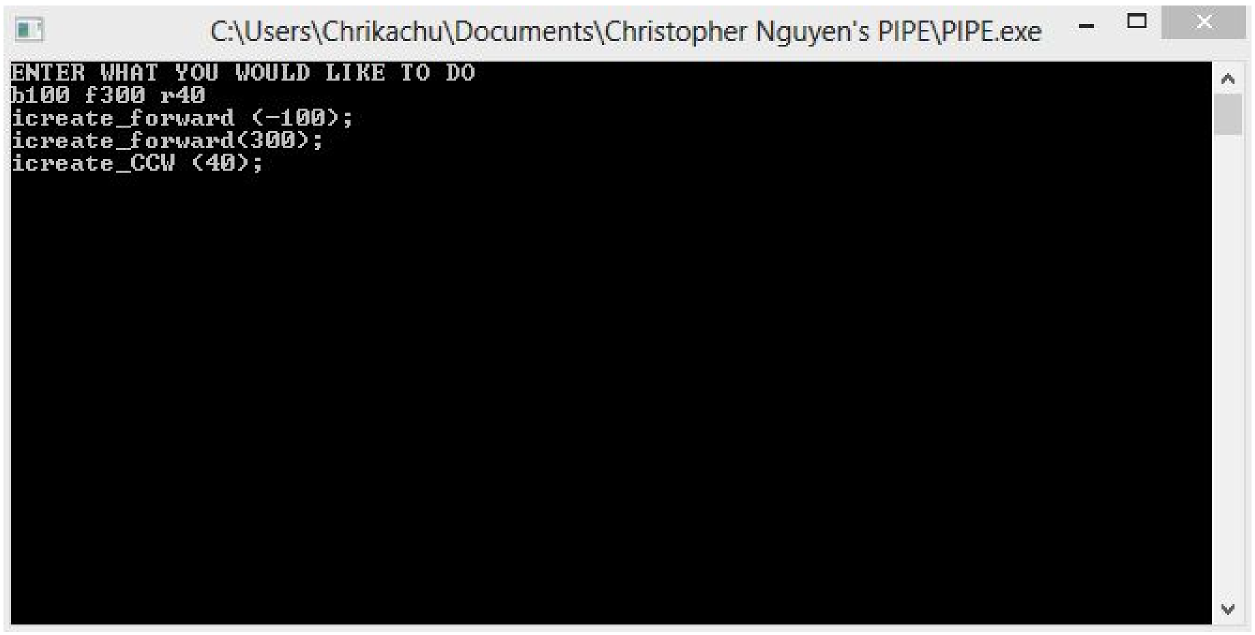
The purpose of this test was to see if it was possible to give an output when given an input. Since character arrays were not used early in the project, words like “forward” was represented with the letter “f.” This program works with commands such as “forward”, “backwards”, and “turn.” This program also saved the output in a .txt. file. In the example, the commands were to go backwards for 100 m, forward for 300 m, then rotate/turn 40-degrees.

First Preliminary Input


C:\Users\Chrikachu\Documents\Christopher Nguyen's PIPE\PIPE.exe

```
ENTER WHAT YOU WOULD LIKE TO DO
b100 f300 r40
```

First Preliminary Output


C:\Users\Chrikachu\Documents\Christopher Nguyen's PIPE\PIPE.exe

```
ENTER WHAT YOU WOULD LIKE TO DO
b100 f300 r40
icreate_forward (-100);
icreate_forward(300);
icreate_CCW (40);
```

FIRST PRELIMINARY TEST CODE

```
#include <stdio.h>

int main()

{

            int yes;

            char c;

            FILE *file_pointer;

            printf("ENTER WHAT YOU WOULD LIKE TO DO\n");

            file_pointer = fopen("Botball_Code.txt", "w");

while (c!= 'z')  {

        scanf(" %c", &c);

            if (c != ' '){

                scanf(" %d", &yes);

        if (c == 'f')

        fprintf (file_pointer, "icreate_forward(%d);\n", yes);

        if (c == 'b')

        fprintf(file_pointer, "icreate_forward (-%d);\n", yes);

        if (c == 'r')

        fprintf(file_pointer, "icreate_CCW (%d);\n", yes);

        }

            else

        scanf("%c", &c);
}
    fclose(file_pointer);
    return 0;
}
```

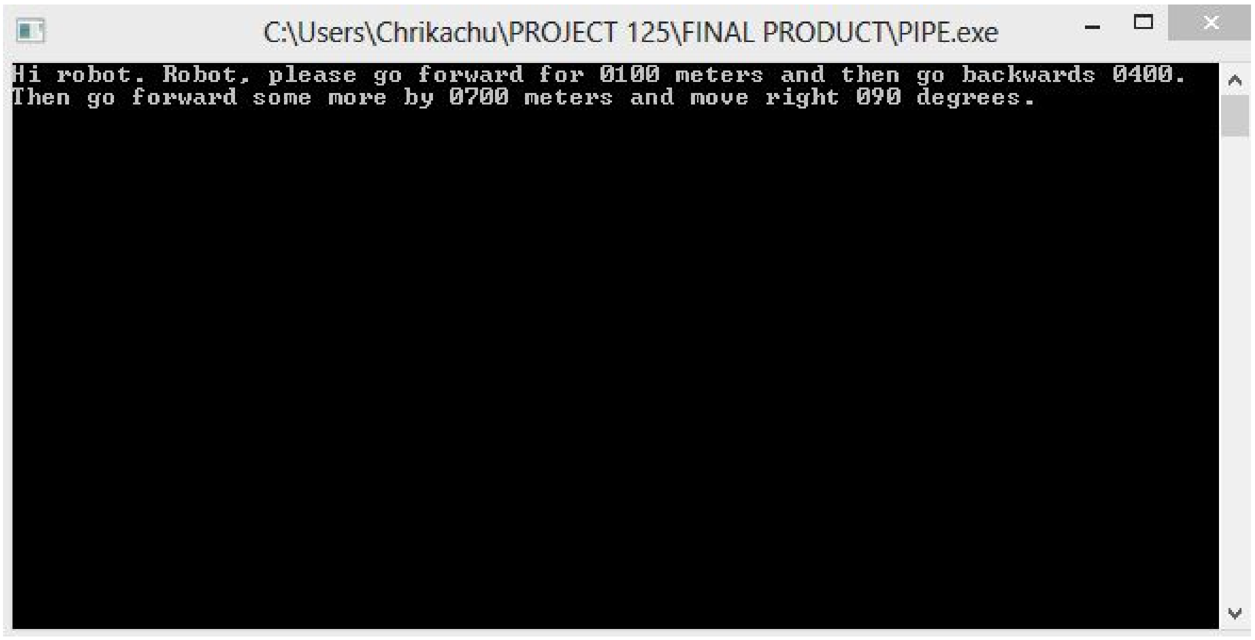
FINAL PRODUCT

```
#include <stdio.h>


main(){

char forward[7] = "forward";                                    //INITIALIZING
RECOGNITION STRINGS

char turn[4] = "turn";

char left[4] = "left";

char right[5] = "right";

char back [4] = "back";

   int yes;

         FILE *file_pointer;

         file_pointer = fopen("PIPE_Botball_Code.txt", "w");

   fprintf (file_pointer, "int main(){\n create_connect();\n");   // OPENS THE.TXT

                                                                //THIS PRINTS int_main() {

                                                                //create_connect();


char word[12];                                                  //INITIALIZING VARIABLES

char read = ' ';

int i, a;


while (1){
    read = getchar();                                    //GRAB LETTER

    if(read == 'f'){                                     //IF IT STARTS WITH F THEN FIGURE
OUT IF ITS TRYING TO SPELL FORWARD
        i = 1; word[0] = 'f';

        while(read != '\n' && read != '.' && read != ' ' && read != ','){       //GRAB REST OF
WORD
            read = getchar();
            word[i] = read;
            i++;}

        a=0; i=0;

        while(i<7){
            if(word[i]!=forward[i])          //CHECKS IF IT IS FORWARD
                a++;
            i++;}

        if(!a){                                      //SKIP TILL NUMBER
            read = getchar();
            while(read != '0')
                read = getchar();
          scanf(" %d", &yes);
          fprintf (file_pointer, "icreate_forward(%d);\n msleep(500);\n", yes); //PRINT FOWARD
COMMAND
          }
    }

    if(read == 'b'){                //IF IT STARTS WITH B THEN FIGURE OUT IF ITS TRYING TO
SPELL BACK
        i = 1; word[0] = 'b';

        while(read != '\n' && read != '.' && read != ' ' && read != ','){       //GRAB REST OF
WORD
            read = getchar();
            word[i] = read;
            i++;}

        a=0; i=0;

        while(i<4){
            if(word[i]!=back[i])            //CHECKS IF IT IS BACK
            a++;
          i++;}

        if(!a){                                        //SKIP TILL NUMBER
          read = getchar();
            while(read != '0')
              read = getchar();
          scanf(" %d", &yes);
          fprintf (file_pointer, "icreate_forward(-%d);\n msleep(500);\n", yes); //PRINT BACK
COMMAND
          }
    }
    if(read == 'l'){                //IF IT STARTS WITH L THEN FIGURE OUT IF ITS TRYING TO
SPELL LEFT
        i = 1; word[0] = 'l';

        while(read != '\n' && read != '.' && read != ' ' && read != ','){       //GRAB REST OF
WORD
          read = getchar();
          word[i] = read;
          i++;}
        a=0; i=0;


        while(i<4){
          if(word[i]!=left[i])            //CHECKS IF IT IS LEFT
            a++;
          i++;}


        if(!a){                           //SKIPS TILL NUMBER
          read = getchar();
            while(read != '0')
              read = getchar();
          scanf(" %d", &yes);
          fprintf(file_pointer, "icreate_CCW (%d);\n msleep(500);\n", yes); //PRINT LEFT
    COMMAND
          }
      }


      if(read == 'r'){               //IF IT STARTS WITH R THEN FIGURE OUT IF ITS TRYING TO
    SPELL RIGHT
        i = 1; word[0] = 'r';


        while(read != '\n' && read != '.' && read != ' ' && read != ','){       //GRAB REST OF
```

```
WORD

            read = getchar();

            word[i] = read;

            i++;}


        a=0; i=0;


        while(i<5){

            if(word[i]!=right[i])            //CHECKS IF IT IS RIGHT

                a++;

            i++;}


        if(!a){

            read = getchar();                  // SKIP TILL NUMBER

                while(read != '0')

                    read = getchar();

            scanf(" %d", &yes);

            fprintf(file_pointer, "icreate_CW (%d);\n msleep(500);\n" , yes); //PRINT RIGHT
COMMAND

            }

    }
    if(read == 't'){                //IF IT STARTS WITH t THEN FIGURE OUT IF ITS TRYING TO
SPELL TURN
        i = 1; word[0] = 't';

        while(read != '\n' && read != '.' && read != ' ' && read != ','){      //GRAB REST OF
WORD
            read = getchar();
            word[i] = read;
            i++;}

        a=0; i=0;

        while(i<4){
            if(word[i]!=turn[i])            //CHECKS IF IT IS TURN
                a++;
            i++;}

        if(!a){
            read = getchar();               //SKIP TILL NUMBER
                while(read != '0')
                    read = getchar();
            scanf(" %d", &yes);
            fprintf(file_pointer, "icreate_CCW (%d);\n msleep(500);\n", yes); //PRINT TURN
COMMAND
            }
    }
    if(read == 'z'){
        fprintf(file_pointer, "}\n", yes); //IF USER TYPES Z CLOSE TXT FILE AND END
PROGRAM
        fclose(file_pointer);
        return 0;
        break;
    }
}
fprintf(file_pointer, "}\n", yes);
fclose(file_pointer);
    return 0;


}
```

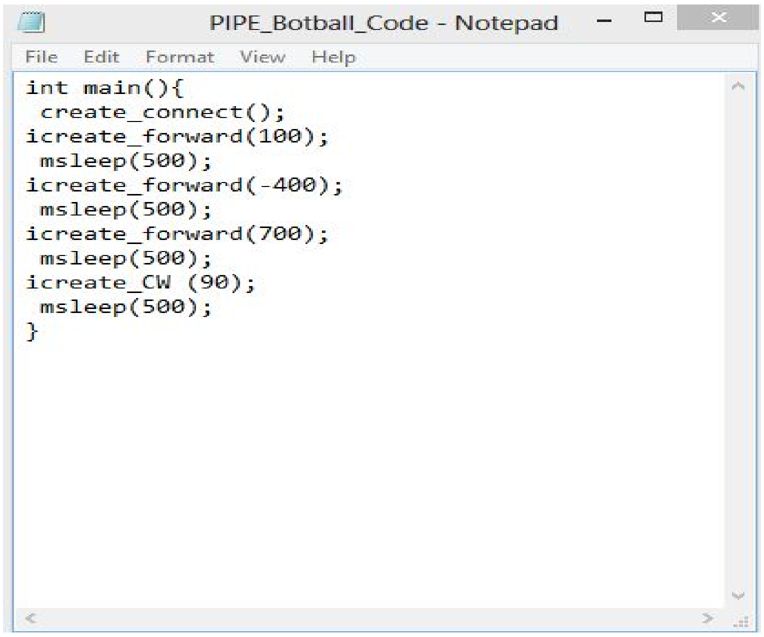
Final Product Sample

English-language Input:



```
Hi robot. Robot, please go forward for 0100 meters and then go backwards 0400.
Then go forward some more by 0700 meters and move right 090 degrees.
```

Code-Output:



```
int main(){
 create_connect();
icreate_forward(100);
 msleep(500);
icreate_forward(-400);
 msleep(500);
icreate_forward(700);
 msleep(500);
icreate_CW (90);
 msleep(500);
}
```

The final product effectively converts the commands in the Botball C programming language.

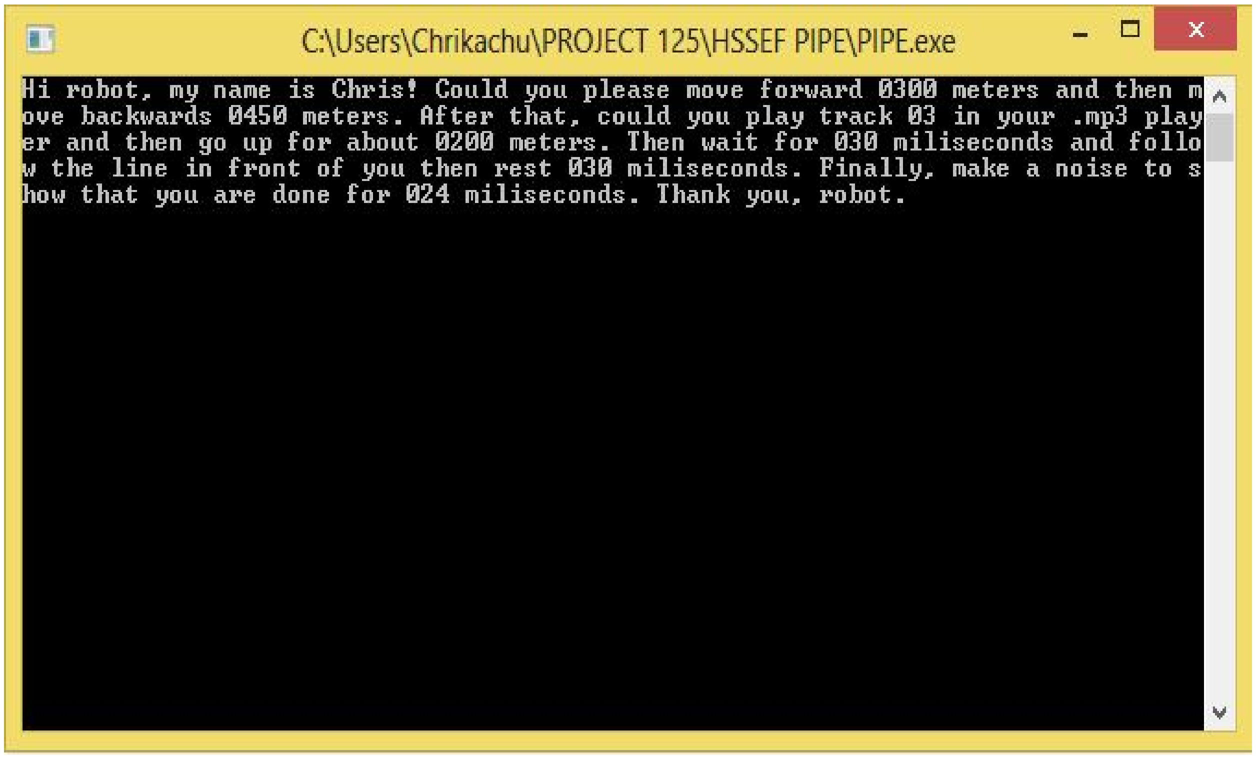
2014 Honolulu District Science Fair

2014 District Science Fair Best of Category in Computer Science

2014 District Science Fair State Science Fair Qualification

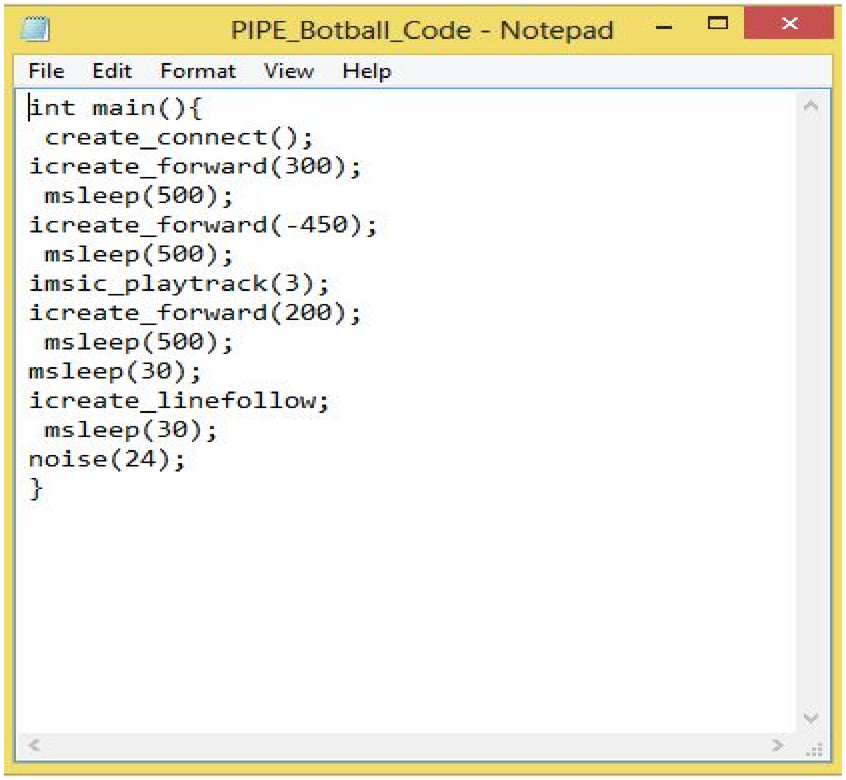
Sample Inputs and Outputs

Input 1:



```
Hi robot, my name is Chris! Could you please move forward 0300 meters and then m
ove backwards 0450 meters. After that, could you play track 03 in your .mp3 play
er and then go up for about 0200 meters. Then wait for 030 miliseconds and follo
w the line in front of you then rest 030 miliseconds. Finally, make a noise to s
how that you are done for 024 miliseconds. Thank you, robot.
```

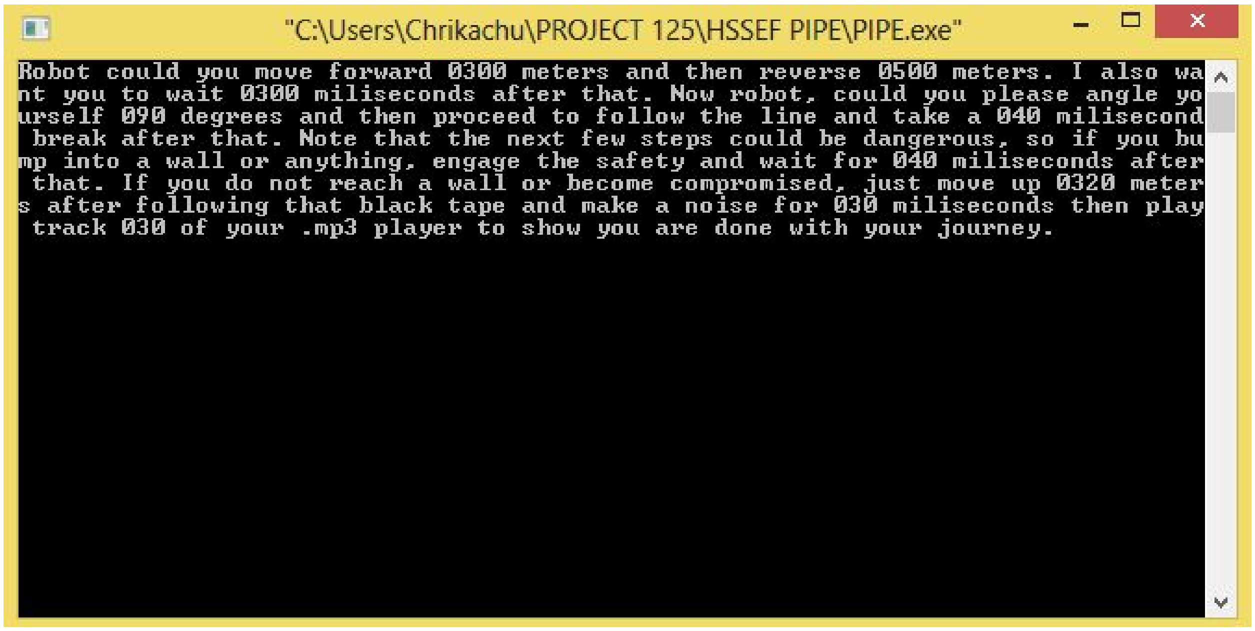
Output 1:



```
int main(){
 create_connect();
icreate_forward(300);
 msleep(500);
icreate_forward(-450);
 msleep(500);
imsic_playtrack(3);
icreate_forward(200);
 msleep(500);
msleep(30);
icreate_linefollow;
 msleep(30);
noise(24);
}
```

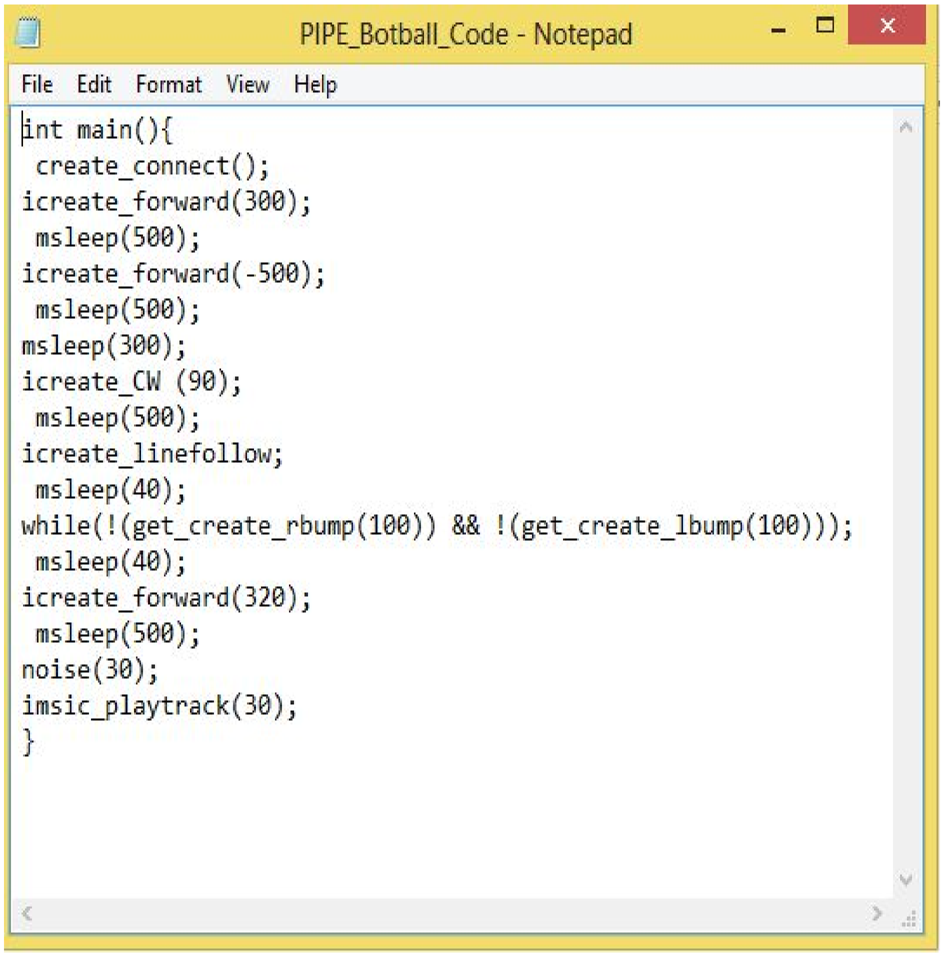
In essence, it is possible for the application to translate many commands in several series of sentences and perform tasks using hardware such as line sensors and .mp3 players.

Input 2:



```
Robot could you move forward 0300 meters and then reverse 0500 meters. I also wa
nt you to wait 0300 miliseconds after that. Now robot, could you please angle yo
urself 090 degrees and then proceed to follow the line and take a 040 milisecond
 break after that. Note that the next few steps could be dangerous, so if you bu
mp into a wall or anything, engage the safety and wait for 040 miliseconds after
 that. If you do not reach a wall or become compromised, just move up 0320 meter
s after following that black tape and make a noise for 030 miliseconds then play
 track 030 of your .mp3 player to show you are done with your journey.
```

Output 2:



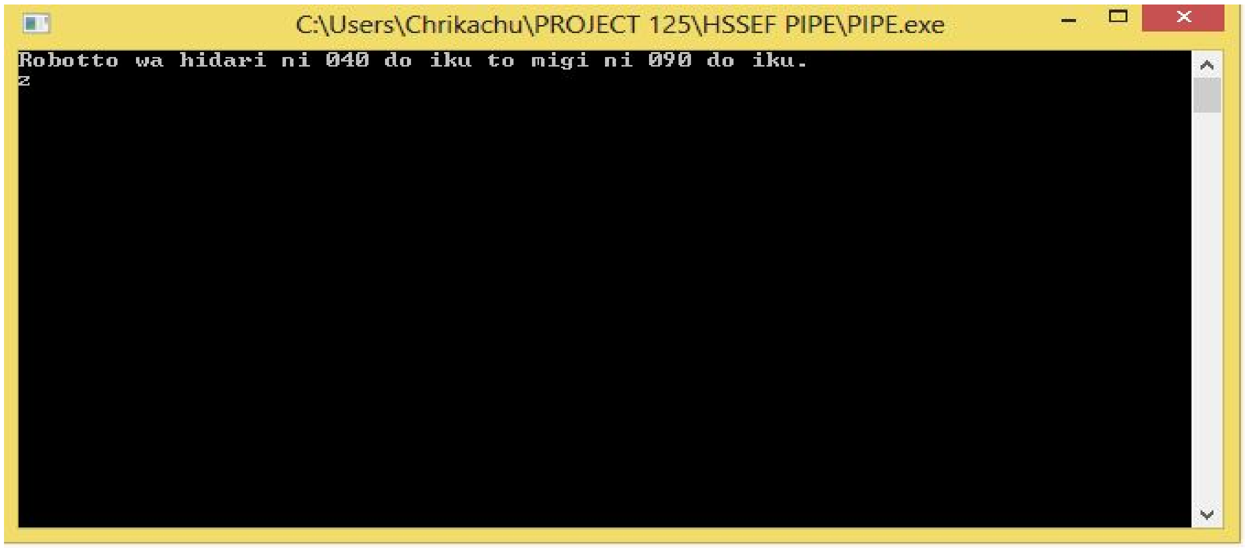
```
int main(){
 create_connect();
icreate_forward(300);
 msleep(500);
icreate_forward(-500);
 msleep(500);
msleep(300);
icreate_CW (90);
 msleep(500);
icreate_linefollow;
 msleep(40);
while(!(get_create_rbump(100)) && !(get_create_lbump(100)));
 msleep(40);
icreate_forward(320);
 msleep(500);
noise(30);
imsic_playtrack(30);
}
```

The application also proves that it is possible to translate commands dependent on loops such as while or if loops. In this scenario, if the bumpers of the robot is touched due to a collision, the robot would realign itself and move back 100 and wait several seconds.

Input 3:

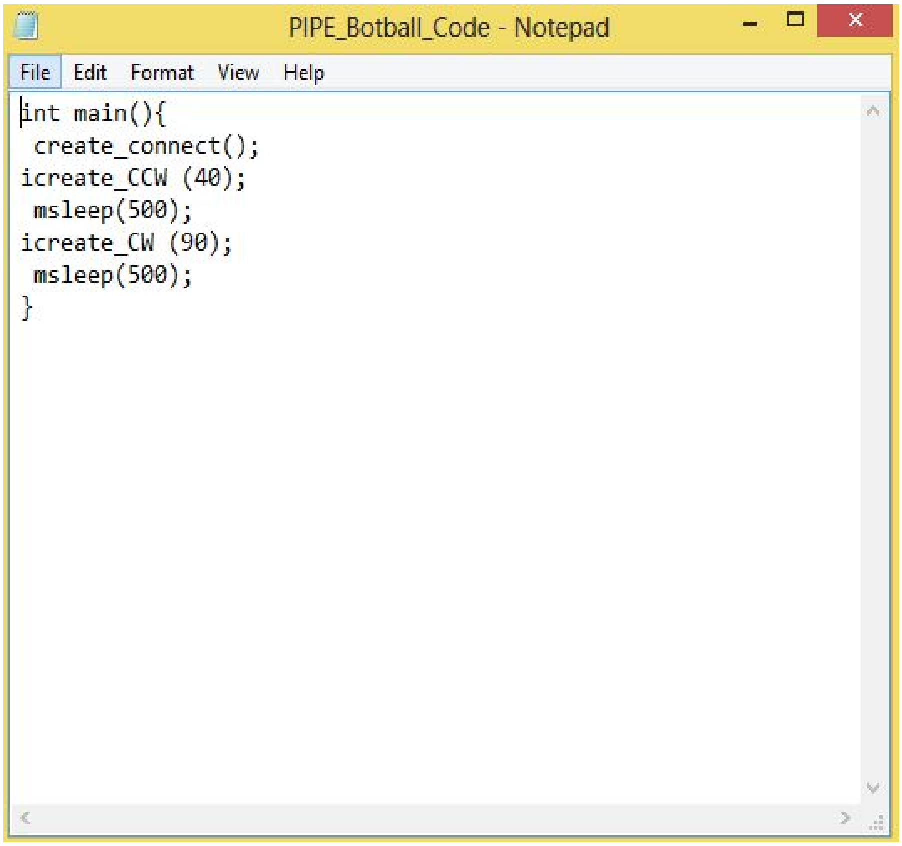
ロボットは左に040度行くと右に090度行く。

Robotto wa hidari ni 040-do iku to migi ni 090-do iku.



```
Robotto wa hidari ni 040 do iku to migi ni 090 do iku.
z
```

Output 3:



```
int main(){
 create_connect();
icreate_CCW (40);
 msleep(500);
icreate_CW (90);
 msleep(500);
}
```

Not only does PIPE translate English-language sentences, but it could also translate Japanese-language sentences when written in roman text or *romaji*. If this concept were to be extended upon, it is possible that not only English-language is translated but almost every language when expressed in roman text.