# **Title:** LOGO Interpreter Interface with Omnibot

**Author:** Raman Sharma, H. Abhinav Gokari

## Introduction:

In this project, we take the standard 'ucblogo' implementation of LOGO interpreter and interface it with the firebird omni-directional bot. LOGO is a complete programming language with power equal to LISP, but it is mainly used to draw graphics and has very intuitive and easy commands to draw graphics. We had to create a system such that given a LOGO program the bot draws the same figure on a paper as the program directs.

1. The LOGO program will be given to the interpreter on a computer (or laptop).
2. The interpreter has to send messages to the robot via zigbee.
3. The robot has to correctly respond to the messages and draw the figure.

## Description:

1. We stripped down the LOGO interpreter to just interpreter and removed any graphics related part (as far as we could). This was done so that we can write the zigbee communication code where it used to draw graphics.

2. We created the basic movement code for the omnidirectional bot.

   We created the following functions for this purpose:

   i. forwardA(), forwardB(),forwardC() [ These functions cause the robot to move in the direction of wheels A,B and C respectively]
   ii. backA(),backB(),backC()[ These functions cause the robot to move opposite to the direction of wheels A,B and C respectively]
   iii. stop() [This function is to stop the motors]
   iv. linear_distance_mm(unsigned char) [it is the waiting function for the robot to travel a certain distance.]
   v. rotate_left(),rotate_right() [These are to rotate the robot ccw and cw respectively ]

vi. angle_rotate(unsigned char) [it is the waiting function for the robot to rotate a certain angle.]

3. We wrote the zigbee communication system and designed the protocol for sending information to the robot on how to move.

i. The protocol has the following structure:
ii. Zigbee allows us to send only one byte of data at a time.
iii. We followed the system of having the first 3 bits as the command and the last 5 bits as the value. i.e. The byte to be sent is <xxxyyyyy> where xxx is the command.
iv. The commands are the following.

| **The 3 bits <xxx>** | **Meaning** | **Special** |
|---|---|---|
| 000 | Orient the robot to the following absolute angle and the value following is to be treated as positive. | We are sending the absolute angle because robot's shape allows it to vary from -30 to +30. If we were sending change we would have required more bits. But, it also means that the robot has to remember its angle. |
| 001 | Orient the robot to the following absolute angle and the value following is to be treated as negative. | |
| 010 | Forward A | Value =0 is never sent (an empty command for future use)<br>Values>32 are broken down in more than one forward<br>command. e.g. forward 60 <=> forward 32 and then forward 28 |
| 011 | Backward A | Value =0 is never sent (an empty command for future use) Values>32 are broken down in more than one forward command. e.g. forward 60 <=> forward 32 and then forward 28 |
| 100 | Forward B | Value =0 is never sent (an empty command for future use) Values>32 are broken down in |

| | | |
|---|---|---|
| | | more than one forward command. e.g. forward 60 <=> forward 32 and then forward 28 |
| 101 | Backward B | Value =0 is never sent (an empty command for future use) Values>32 are broken down in more than one forward command. e.g. forward 60 <=> forward 32 and then forward 28 |
| 110 | Forward C | Value =0 is used to signify that the next movement is to be a penup movement and hence no<br>line should be drawn. Values>32 are broken down in more than one forward command. e.g. forward 60 <=> forward 32 and then forward 28 |
| 111 | Backward C | Value =0 is never sent (an empty command for future use) Values>32 are broken down in more than one forward command. e.g. forward 60 <=> forward 32 and then forward 28 |

4. To rotate the robot, we have to rotate it about the pen which is at the back of robot, so we take the robot a bit back and rotate it about the the center(which is the only possibility) and then move forward.

## Installation instruction for ucblogo:

1. Required Library : libtermcap (rest are normally available on any linux OS)
2. Extract the files in ucblogo.tar.gz to a folder.
3. Go in the folder and type 'make'.
4. The executable 'logo' is the required executable (Note: to run it you need root
5. permission as it will be writing on a USB port)

## References:

Firebird V AVR 2560 Software manual
Firebird V AVR 2560  Hardware Manual
Firebird Omnidirectional Bot manual
http://www.comptechdoc.org/os/linux/programming/c/linux_pgcserial.html