Robotics

Behavior-based Fire Alarm Robot

By, Rohit Santosh Nagvenkar

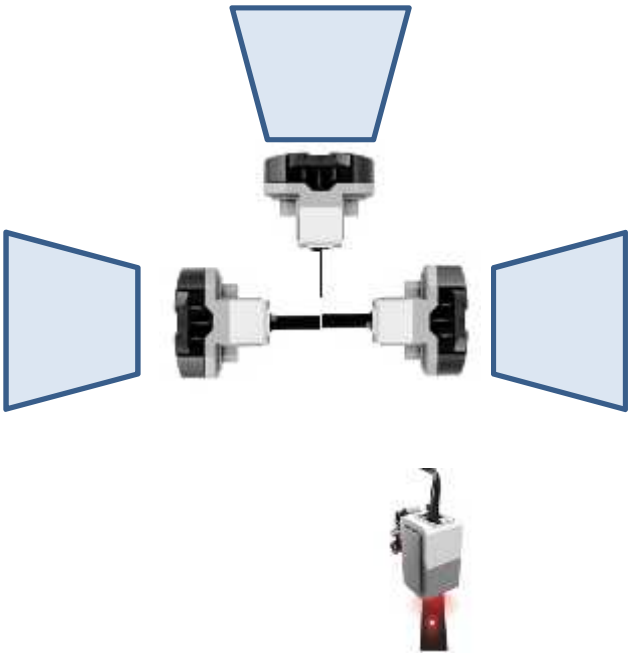
Robotics

Contents

Robotics

Design:

Implementation:



There are 4 modes that have been implemented within the system.

1. Wandering
2. Wall Following
3. Searching
4. Extinguish

Wandering:

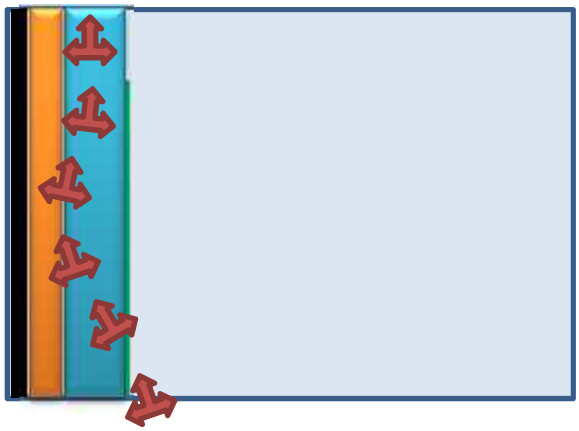
The moment you activate the bot, it goes into wandering mode basically this checks to see if the front, left and right sonars have anything in front of them. The threshold value selected for this is 20cms; choice of such a high value is considering the size of the bot and the way our rotate function works.

When the threshold is not being breached, it moves Forward 90% of the time with a pinch of 5% of either Left turns and 5% of Right turns in best case scenarios. All of this has been implemented using a randomizer and priority has been given to Forward movement this is considering the task for the project is inside a closed environment. So it has to hit a wall at some point and the best way to reach our goal is by moving forward at full speed.

Once the threshold has been breached, the bot knows that there is going to be a wall/obstacle in front of it, and the next course of action is to avoid hitting that wall / obstacle. It does this by randomly selecting whether to go in the left or the right direction once the randomizer comes

up with its answer; it activates the required function to go in the direction it chooses to. Why randomly? This is because the bot has no idea where the flame is.

Wall Following:



The solid black line represents the wall, 3D light brown bar represents the inner threshold, the 3D darker blue block (nominal threshold) represents the path the bot will be wall following on and the lighter blue region represents the outer region.

Let's explain this with an example, the bot starts off in wandering mode where it just moves around wanting to hit something and whens it does hit something, it either makes a right turn or left we don't care about that all we care about is that now it knows there is a wall and it needs to follow it just like how people behave when the power goes off (touching the wall looking for the flash light).

Here comes the threshold values, the 1st one is the inner threshold and once breached, it makes either left or right turn depending on which side of the wall it is on and tries to move itself within the nominal threshold.

Searching:

While the bot is switching between wall following and wandering mode, it continuously looks for light intensity values of the floor using a light sensor. We know the floor is black; the reading for this on the sensor is 1 and we also know the flame is on a bluish colored tile; the reading is 2 on the sensor. This makes our job easier and all the bot has to do now is stop moving around when it sees that colored tile.

Extinguish

Once it's done with searching for the flame (looking at the tile color using a light sensor), the bot now goes into extinguish mode. Here the front and rear wheels of the bot rotate in opposite direction of each other and both motors get powered on at full speed (throttle), this action makes the bot spin around very fast and blow the candle out using air (that's what we hoped it would do but nothing, even at full throttle it's not that fast to generate wind turbulence).

Code:

```
#include <ev3.h>
#include <ev3_lcd.h>
#include <stdio.h>
#include <ev3.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

#define FIX_ANGLE 4
#define CHECK_ANGLE 2
#define ROTATE_RIGHT 1
#define ROTATE_LEFT -1
#define FORWARD_ONE_SQR 8
#define FORWARD_HALF_SQR 4
#define NORMAL_SPEED 19
#define SONAR_RANGE_MIN 25
#define SONAR_FOLLOW_RANGE 5
#define Wall_THRESHOLD 20

void turnRearMotor(int ROTATION, int DIRECTION);
```

```
void turnFrontMotor(int ROTATION, int DIRECTION);
void rightTurn(void);
void leftTurn(void);
void spin(void);
void search(void);
void forward(int y, int speed);
void forward_LIGHT(int y, int speed,int light);
void sonarSTUFF(int sonar_FRONT,int sonar_LEFT,int sonar_RIGHT);
void Truewander();
void rndTurn();
void fix(int sonar_LEFT,int sonar_RIGHT);

void rightTurn(void)
{
	turnFrontMotor(45,ROTATE_RIGHT);
	turnRearMotor(45,ROTATE_LEFT);

	//fix(error, error1);
	forward(1,NORMAL_SPEED);

	turnFrontMotor(45,ROTATE_LEFT);
	turnRearMotor(45,ROTATE_RIGHT);
}

void leftTurn(void)
{
	turnFrontMotor(45,ROTATE_LEFT);
	turnRearMotor(45,ROTATE_RIGHT);

	forward(1,NORMAL_SPEED);

	turnFrontMotor(45,ROTATE_RIGHT);
	turnRearMotor(45,ROTATE_LEFT);

}
void spin(void)
{
	Off(OUT_AD);
	turnFrontMotor(90,ROTATE_RIGHT);
	turnRearMotor(90,ROTATE_LEFT);
	forward(5,50);
	Off(OUT_AD);
	turnFrontMotor(90,ROTATE_LEFT);
	turnRearMotor(90,ROTATE_RIGHT);
	Off(OUT_AD);

}
void search(void)
{
	Off(OUT_AD);
```

```
	RotateMotor(OUT_C,-30,450);
	Wait(SEC_2);
	RotateMotor(OUT_C,30,900);
	Wait(SEC_2);
	RotateMotor(OUT_C,-30,470);
	Wait(SEC_2);

}
void forward(int y,int speed)
{
	y = y*125;
	OnFwdSync(OUT_AD,speed);
	while (0 <= y)
	{
		y = y - 1;
		Wait(1);
	}
}


void backward(int y,int speed)
{
	Off(OUT_AD);
	y = y*125;
	OnRevSync(OUT_AD,speed);
	while (0 <= y)
	{
		y = y - 1;
		Wait(1);
	}
	Off(OUT_AD);
}

void turnRearMotor(int ROTATION, int DIRECTION)
{
	ROTATION = ROTATION * 5;// 1 = 5 degrees
	RotateMotor(OUT_B,50*DIRECTION,ROTATION);
	//450 is 90 degrees
	//1800 is full rotation
}
void turnFrontMotor(int ROTATION, int DIRECTION)
{
	ROTATION = ROTATION * 5;// 1 = 5 degrees
	RotateMotor(OUT_C,50*(-DIRECTION),ROTATION);
	//450 is 90 degrees
	//1800 is full rotation
}

int randomgenerator(){
	time_t t;
```

```
	srand((unsigned) time(&t));
	return rand() % 10;
}

void forward_LIGHT(int y,int speed,int light)
{
	int sonar_FRONT = readSensor(IN_1);
	int sonar_LEFT = readSensor(IN_2);
	int sonar_RIGHT = readSensor(IN_3);
	char mp = MotorPower(OUT_AD);
	y = y*125;
	OnFwdSync(OUT_AD,speed);
	while (0 <= y)
	{
		sonar_FRONT = readSensor(IN_1);
		sonar_LEFT = readSensor(IN_2);
		sonar_RIGHT = readSensor(IN_3);
		if(sonar_FRONT > SONAR_RANGE_MIN){
			fix(sonar_LEFT,sonar_RIGHT);
		}
		light = readSensor(IN_4);
		y = y - 1;
		if(light == 2){
			y = -1;
			Off(OUT_AD);
		}
		Wait(1);
	}

}

void Truewander(){
	if(randomgenerator() == 0){
		rightTurn();
	}else if(randomgenerator() == 1){
		leftTurn();
	}else {
		forward(FORWARD_ONE_SQR,NORMAL_SPEED);
	}
}
void rndTurn(){
	if(randomgenerator() < 5)
	{
		rightTurn();
	}else {
		leftTurn();
	}
}
```

```
void sonarSTUFF(int sonar_FRONT,int sonar_LEFT,int sonar_RIGHT){
	if(sonar_FRONT < SONAR_RANGE_MIN && sonar_LEFT  > SONAR_RANGE_MIN && sonar_RIGHT
> SONAR_RANGE_MIN){
		rndTurn();
	}
	else if(sonar_FRONT < SONAR_RANGE_MIN && sonar_RIGHT < SONAR_RANGE_MIN){
		leftTurn();
	}
	else if(sonar_FRONT < SONAR_RANGE_MIN && sonar_LEFT < SONAR_RANGE_MIN){
		rightTurn();
	}else if(sonar_FRONT < SONAR_RANGE_MIN && sonar_LEFT  < SONAR_RANGE_MIN &&
sonar_RIGHT < SONAR_RANGE_MIN){
		backward(FORWARD_ONE_SQR,NORMAL_SPEED);
		forward(FORWARD_HALF_SQR,NORMAL_SPEED);
		rndTurn();
	}
}
void fix(int sonar_LEFT,int sonar_RIGHT){

	if(sonar_LEFT > SONAR_FOLLOW_RANGE && sonar_LEFT < SONAR_FOLLOW_RANGE + 4){
		//left
		turnFrontMotor(Wall_THRESHOLD, ROTATE_LEFT);
		forward(1 , NORMAL_SPEED);
		turnFrontMotor(Wall_THRESHOLD, ROTATE_RIGHT);
	}else if(sonar_LEFT < SONAR_FOLLOW_RANGE){
		turnFrontMotor(Wall_THRESHOLD, ROTATE_RIGHT);
		forward(1 , NORMAL_SPEED);
		turnFrontMotor(Wall_THRESHOLD, ROTATE_LEFT);
	}
	if(sonar_RIGHT > SONAR_FOLLOW_RANGE && sonar_RIGHT < SONAR_FOLLOW_RANGE + 4){
		//right
		turnFrontMotor(Wall_THRESHOLD, ROTATE_RIGHT);
		forward(1 , NORMAL_SPEED);
		turnFrontMotor(Wall_THRESHOLD, ROTATE_LEFT);
	}else if(sonar_RIGHT < SONAR_FOLLOW_RANGE){
		turnFrontMotor(Wall_THRESHOLD, ROTATE_LEFT);
		forward(1, NORMAL_SPEED);
		turnFrontMotor(Wall_THRESHOLD, ROTATE_RIGHT);
	}
}


int main(void)
{
	InitEV3();
	setAllSensorMode(US_DIST_CM,US_DIST_CM,US_DIST_CM,COL_COLOR);
	int sonar_FRONT = 0;
	int sonar_RIGHT = 0;
	int sonar_LEFT = 0;
	int light = 0;
```

```
	int breaker = 0;
	LcdInit();
	while(breaker == 0){
		sonar_FRONT = readSensor(IN_1);
		sonar_LEFT = readSensor(IN_2);
		sonar_RIGHT = readSensor(IN_3);
		light = readSensor(IN_4);
		if(light == 2)
		{
			breaker = 1;
		}else{
			forward_LIGHT(1 , NORMAL_SPEED,light);
			sonarSTUFF(sonar_FRONT,sonar_LEFT,sonar_RIGHT);
		}
	}
	FreeEV3();
	return 0;
}
```