DEPAUL UNIVERSITY
COLLEGE OF COMPUTING AND DIGITAL MEDIA

unesp

CSC 375/475: Introduction to Robotics - Spring 2019/20

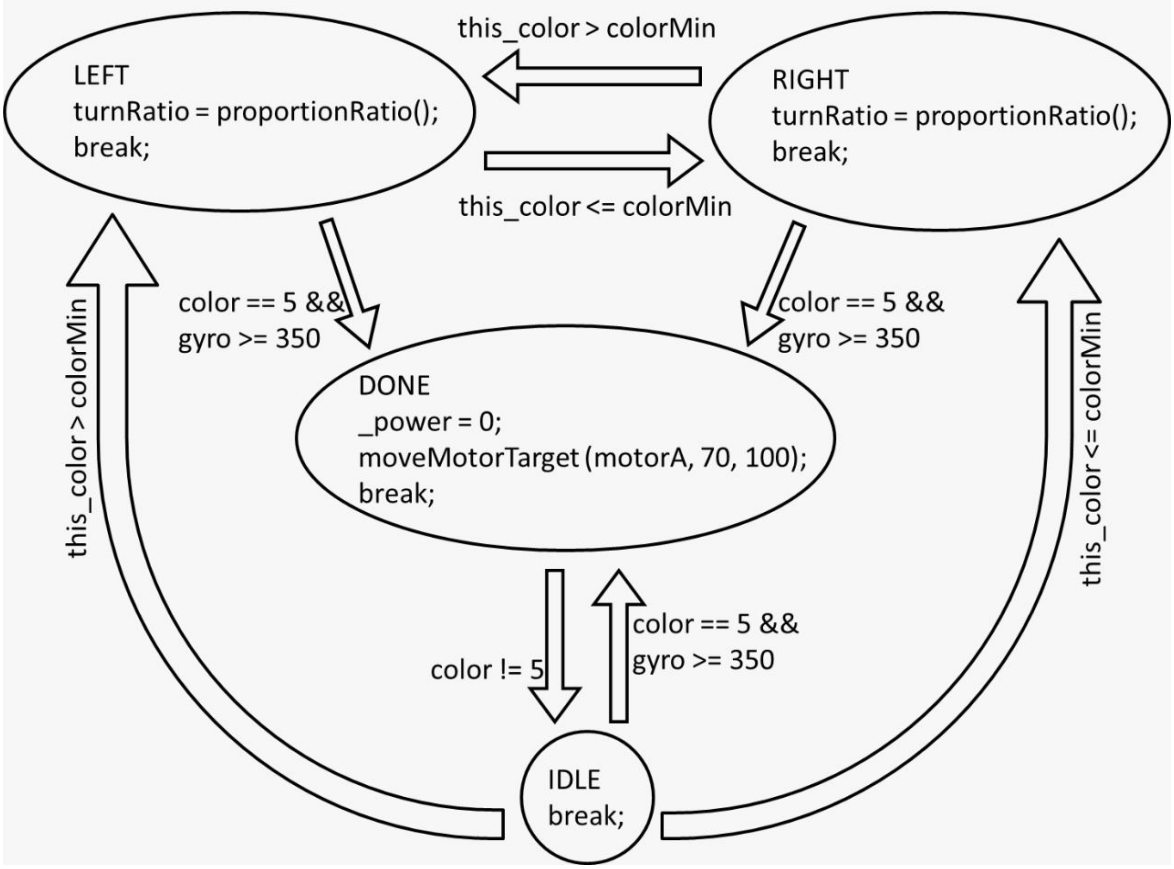
Participants	%Contribution	University
Willian Ibanez Mesquita	20	Unesp
Pius Colletti Oehling	20	Unesp
Luis Guilherme Fernandes Pereira	10	Unesp
Karen de Souza Gomes	10	Unesp
William	20	DePaul
Seth	20	DePaul

GLE Competition 1: Line Tracking
Group 6

05/20/2020

Exercise: Line Tracking Challenge

Q.A) Finite State Diagram



- This_color is the color reflected to the color sensor (S3) the colorMin equals to 45 it's the arithmetic mean of the values corresponding to the black and white colors.
- Color is the color is the color name got from the color sensor (getColorName(S3)) and gyro it's the globalGyroValue (got from gyro sensor).

Right to left: When this_color > colorMin the robot will turn to the left.

Left to right: When this_color <= colorMin the robot will turn to the right.

Right to done: When color == 5 (red) and gyro >= 350 the robot will stop, and will lift his arm releasing the box in the red square.

Left to done: When color == 5 (red) and gyro >= 350 the robot will stop, and will lift his arm releasing the box in the red square.

Idle to right: When this_color <= colorMin the robot will enter the state right.

Idle to left: When this_color > colorMin the robot will enter the state left.

Idle to done: When color == 5 (red) and gyro >= 350 the robot will stop, and will lift his arm releasing the box in the red square.

Done to idle: if the color is different from 5 (red) the robot will enter the state idle.

Q.B) Code, Pseudocode and Refresh Rates

CODE:

```
enum STATES{IDLE=0, LEFT=1, RIGHT=2, DONE=3};
enum STATES state = IDLE;
     //light
int init_startD = 90;
     //dark
int init_startC = 0;
     //minimum threshold
float colorMin = (init_startC + init_startD) / 2;
     //gyroscope counter
float globalGyroValue = 0;
     //reflectance of the floor
double this_color = -1;
     //color sensor refresh rate
int colorSenseDelay = 200 / 1000;
     //motor power throughout the line tracking
const float start_motor_power = 45;
int _power = start_motor_power;
     //turn ratio
const double beginningTurn_ratio = 75;
double turnRatio = beginningTurn_ratio;
     //motors refresh rate
int controller_motor_delay = 500 / 1000;
     //turn gain
int turn_gain = 1.5;
     //debuger refresh rate
int debug_sleep = 1000 / 5;
     //color sensed
int color;

task senseColor()
{
```

```
	while(true)
	{
		this_color = getColorReflected(S3);
		color = getColorName(S3);
		sleep(colorSenseDelay);
	}
}

task motorControl()
{
	int prev_power = 0;
	int prev_ratio = 0;
	while(true)
	{
		if(prev_power != _power || prev_ratio != turnRatio)
		{
			setMotorSync(motorB , motorC , turnRatio ,
_power);
		}
		prev_power = _power;
		prev_ratio = turnRatio;
		sleep(controller_motor_delay);
	}
}

task debug()
{
	while(true)
	{
		writeDebugStreamLine("current_color=%d\n\t
motor_power=%d\n\t\t motor_turn_ratio=%d\n\t\t state=%d gyro:
%d" , this_color, _power, turnRatio, state, globalGyroValue);
		sleep(debug_sleep);
	}
}

task gyroSensor1()
{
	while(true)
	{
		globalGyroValue = getGyroDegrees(S2);
	}
}

double proportionRatio()
```

```
{
     double discrepancy = this_color - colorMin;
     double out = (turn_gain * discrepancy) / (colorMin *
turn_gain);
     out *= -1 * beginningTurn_ratio;
     return out;
}

task main()
{
          //go forward
     setMotorSync(motorB , motorC , 0 , 75);
          //lower the arm
     moveMotorTarget(motorA, -70, 80);
     sleep(1100);

     clearDebugStream();
     startTask(senseColor);
     startTask(motorControl);
     startTask(debug);
     startTask(gyroSensor1);

     while(true)
     {

          switch(state)
               {
               case IDLE:
                    if(this_color != colorMin) state = LEFT;
                    //if(this_color <= colorMin) state =
RIGHT;
                    if(color==colorRed &&
globalGyroValue>=350) state = DONE;
                    break;
               case LEFT:
                    if(this_color <= colorMin) state = RIGHT;
                    if(color==colorRed &&
globalGyroValue>=350) state = DONE;
                    break;
               case RIGHT:
                    if(this_color > colorMin) state = LEFT;
                    if(color==colorRed &&
globalGyroValue>=350) state = DONE;
                    break;
               case DONE:
```

```
                    break;
            }

            switch(state)
                  {
                  case IDLE:
                        moveMotorTarget(motorA, -70, 80);
                        sleep(2000);
                        _power=0;
                        break;
                  case LEFT:
                        turnRatio = proportionRatio();
                        break;
                  case RIGHT:
                        turnRatio = proportionRatio();
                        break;
                  case DONE:
                        _power=0;
                        moveMotorTarget(motorA, 70, 80);
                        sleep(2000);
                        break;
            }
      }
}
```

PSEUDOCODE:

1) Embrace the box (lower the arm) and simultaneously go forward until black;
2) Start line tracking:
    a) Determine threshold → (white color value + black color value) / 2;
    b) If color reflected is lighter than the threshold, turn left until threshold;
    c) If color reflected is darker than the threshold, turn right until threshold;
3) Proceed with line tracking algorithm until end condition;
4) End condition → color = red;
5) Stop line tracking and simultaneously stop motors;
6) Release the box (rise the arm).

REFRESH RATES:

Initially, the refresh rates were based on values told by the teachers. But since they define how many times a second s sensor, for example, will refresh his output values, they could be adjusted to fit a more stressful situation to the robot. That would be a situation in which the speed of motors is higher than normal and the sensors can't keep up to maintain the robot in track. By increasing the refresh rates, the robot will be more responsive and thus will

be less likely to lose track. Therefore, making possible a use of higher speeds and so completing the track in a shorter time, giving a better result to the group.

Q.C) GLE collaboration

PERSPECTIVE FROM THE DEPAUL STUDENTS:

The collaboration between UNESP and DePaul students has been learning the strengths and weaknesses of each individual and group. The UNESP students have a fond familiarity with engineering concepts and the drawing up of state machines as well as comprehending algorithms to handle dynamically changing values. This was very helpful in allowing the dividing up of work in order to efficiently draw up solutions, while also trusting the validity of those solutions.

The DePaul contributions were mostly in conceiving the RobotC programming syntax to use in order to implement the algorithms correctly. Zoom and WhatsApp communications were used a lot throughout developing the solution and even though there is a 4hour time difference between Chicago and the area of Brazil UNESP is in, we were able to have a productive Zoom video session that allowed real-time DePaul and UNESP student collaboration. The UNESP students are very good at extrapolating the logic from the RobotC programs for design into the state-machines. I believe this process actually helped both student groups understand each side of the assignment more clearly.

PERSPECTIVE FROM THE UNESP STUDENTS:

The collaboration with DePaul students was made mainly by WhatsApp and the Zoom synchronous meetings. The Zoom online classes occurred once a week and messages were constantly sent. Mainly due to language fluency, the DePaul students contributed more during the classes. But during the week, both groups showed improvements and discussions on the assignments. It also looks like the DePaul students had a better understanding of the programming subject, maybe because they had more frequent classes over on their side, that also had some influence on the contribution to new strategies and ideias.