ECE 450: Bumper Car

Building the basic robot

In this exercise we will build a robot with the most basic behavior, escaping whenever it hits something with its bumpers. Perform the following programming steps:

1. Program the START and STOP button so that:
   - Pushing the START button sends the robot forward
   - Pushing the STOP button stops the robot and puts it back into its initial state (i.e., pushing the START button again sends the robot forward).
   - Review code shown in class for the ***RESET*** behavior and the ***Arbitrate*** routine

2. ***Create an ESCAPE behavior so that if a bumper switch is activated:***
   - Backup a quarter of the length of your robot using the IC time commands.

   - Turn 45° in the opposite direction of the bumper switch activated (i.e., turn right if the left bumper switch was engaged) and then continue forward.

You may use the attached outline for your code.

Lab Code:

- Show your code to the Instructor or TA at the beginning of the lab. (plan on leaving a hardcopy)
- Email your final code to the Instructor and TA by midnight on the date of the demo. Be sure to follow the given structure for your code.

Lab Report:

1. Draw a Subsumption Diagram.
2. Draw a State Transition Diagram.
3. Describe at least 2 ways that would ensure a more accurate 45° turn.
4. What could you do to detect the wall before striking it? Are there any shortcomings to your method?
5. Submit a hardcopy of your report at the beginning of the class on the date specified on the course website.

```
// ECE 450: Bumper Car Lab
// Team #
// Names?

/*Port Definitions */
int RIGHT_MOTOR =   0;       /* Motor Slot 0 */
int LEFT_MOTOR  =   2;          /* Motor Slot 2 */

int LEFT_BUMPER =   ?;
…

/* Logic */
#define TRUE  1
#define FALSE 0

/* Motor Commands */
#define REST            0
#define STOP            1
#define FORWARD         2
#define REVERSE         3
#define TURN_RIGHT      4
#define TURN_LEFT       5

/* Global Variables */
int reset_action;
int escape_action;
int escape_time;
…

void main()
{  int motor_

   while(TRUE)
   {
      start_press();

     //Initialize the behaviors
        escape_action = REST;

     //Initialize local variables
      old_motor_cmd = -1;
      flag = TRUE;

      while(flag) //Arbitrate
      {
         //Call Behaviors
         …


        //Arbitration
         motor_cmd = FORWARD; //Default

         if (reset_action != REST)
         {  flag = FALSE;
            motor_cmd = reset_action;
         }
         else if escape_action != REST
         {
           …
         }

         if (motor_cmd != old_motor_cmd)
         {  motor_control(motor_cmd);
            old_motor_cmd = motor_cmd;
         }

      }
   }
} //end of main routine


void escape()
{    …
}



void motor_control(int motor_cmd) // Defines motor actions
{
    If (motor_cmd == FORWARD)
          Set Right and Left motors to go forwards
    If (motor_cmd == REVERSE)
          Set Right and Left motors to go backwards
    If (motor_cmd == TURN_RIGHT_45)
          Set Right and Left motors so robot turns right
    If (motor_cmd == TURN_LEFT_45)
          Set Right and Left motors so robot turns left
    If (motor_cmd == STOP)
          Stop!
}//end of motor_control definition
```