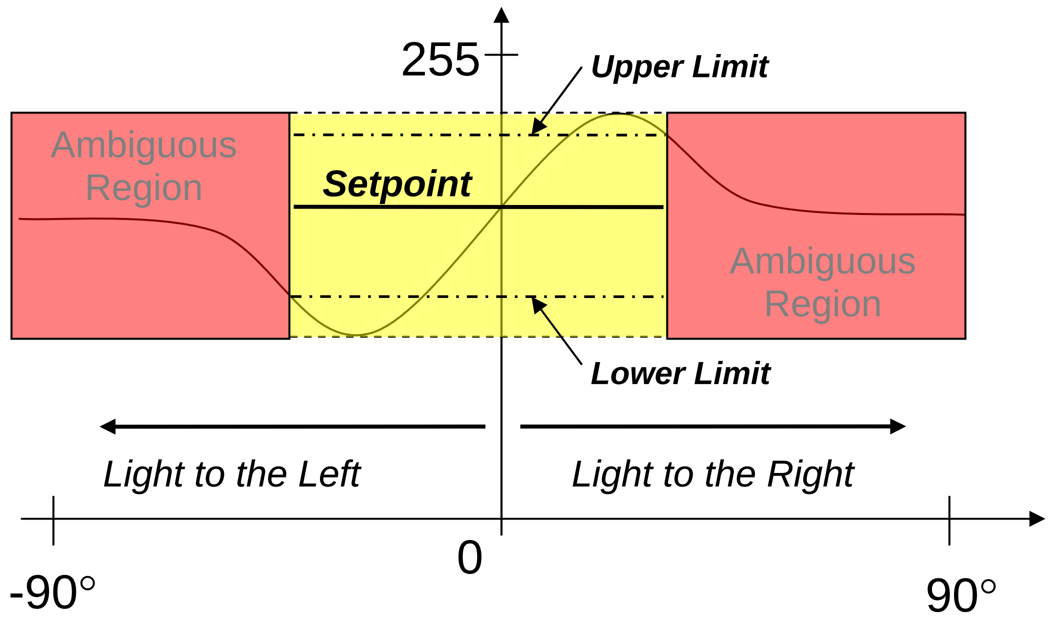
Lab 3: *Search Lab*

I. Overview

In this lab we'll program the robot to us a Proportional – Differential (PD) controller to follow a light. Determine an upper and lower limit for locating the light and an appropriate setpoint based on your Homework results.



Using the upper limit, lower limit and setpoint, create a robot with three behaviors: **Search**, **Terminate** and **Follow**. Under the Search behavior, the robot will search for a bright light. The differential detector system consisting of two CdS photoresistors will be used to detect the bright light (bring a bright flashlight if you can with a large enough beam to illuminate both sensors). Once detected, the robot should move forward (Follow) adjusting its direction of travel until a bumper is struck, then shut down (Terminate) which should close down the robot.

II. Lab Code

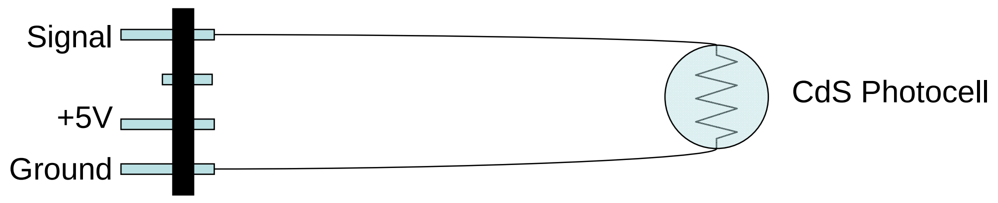
- Show your completed code using the template below to the Instructor or TA at the beginning of the lab period.
- Send your working code to the Instructor and TA by midnight on the date of the demo. Be sure to follow the given structure for your code.

III. Lab Report:

a. Sketch a State Transition Diagram for the Search behavior
b. Sketch a Subsumption Diagram.

c. Compare and contrast the use of a Proportional Controller to the use of a Proportional-Differential Controller (e.g., the use of the velocity term). How significant was the addition of the Differential Controller?

d. In the homework, you calculated a maximum and minimum value the analog() function would give using the two CdS sensors. Using the resistance values you measured for one of these sensors, calculate the range of values the analog() function would return if you connected it to the Handyboard as shown below. Compare the range of values using this single CdS Sensor to those you calculated in the homework exercise using two CdS Sensors.



```
// ECE 450: PD Search Routine
// Team #
// Names?

/*Port Definitions */
int RIGHT_MOTOR =   ?           /* Motor Slot ? */
int LEFT_MOTOR =    ?          /* Motor Slot  ? */
int LEFT_BUMPER =    ?
int RIGHT_BUMPER = ?

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

/* Motor Commands */
#define FORWARD     1
#define REVERSE     2
#define TURN_LEFT   3
#define TURN_RIGHT  4
#define STOP        5
#define FOLLOW      6

/* Global Variables */
   …
    //define global variables
   …
void main()
{  …
    //define local variables
   …


   while(TRUE)
   {
      start_press();

     //Initialize the behaviors
        search_action = REST;
        follow_action = REST;
        terminate_action = REST;

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

      while(flag) //Arbitrate
      {
          //Call Behaviors
        search();
        follow();
        terminate();


         //Arbitration
         motor_cmd = FORWARD; //Default

         if (stop_button())
         {  flag = FALSE;
            motor_cmd = reset_action;
         }
         else if ?_action != REST
         {
           …
         }
            else if ?_action != REST
         {
           …
         }

         else if ?_action != REST
         {
           …
         }

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

      }
   }
} //end of main routine


void search()
{    …
}

void terminate()
{    …
}

void follow()
{    …
}

void motor_controller(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)
            Set Right and Left motors so robot turns right
      If (motor_cmd == TURN_LEFT)
            Set Right and Left motors so robot turns left
      If (motor_cmd == STOP)
            Stop!
      If (motor_cmd = FOLLOW)
            Use the power term to go towards the light!

}//end of motor_ controller definition
```