```c
// Lab2_nmain.c
// Runs on MSP432

/** Name: Kenzie Moore
*  Program: Electrical Engineering Technology
*  Year: 2nd year
*  Class: Microcontroller Systems
*  Section: CPET 253
*  Exercise: Lab 2 Prelab
*  Date : 1/16/2021
*/

/* This example accompanies the book
   "Embedded Systems: Introduction to Robotics,
   Jonathan W. Valvano, ISBN: 9781074544300, copyright (c) 2019
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/

Simplified BSD License (FreeBSD License)
Copyright (c) 2019, Jonathan Valvano, All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are
those of the authors and should not be interpreted as representing official
policies, either expressed or implied, of the FreeBSD Project.
*/
#include <stdint.h>
enum scenario {
    Error = 0,
    LeftTooClose = 1,
    RightTooClose = 2,
    CenterTooClose = 4,
    Straight = 8,
    LeftTurn = 9,
    RightTurn = 10,
    TeeJoint = 11,
    LeftJoint = 12,
    RightJoint = 13,
    CrossRoad = 14,
    Blocked = 15
};
typedef enum scenario scenario_t;

#define SIDEMAX 354    // largest side distance to wall in mm
#define SIDEMIN 212    // smallest side distance to wall in mm
#define CENTEROPEN 600 // distance to wall between open/blocked
#define CENTERMIN 150  // min distance to wall in the front
scenario_t Classify(int32_t Left, int32_t Center, int32_t Right){
  scenario_t result=Error;
  if((Left == SIDEMIN) & (Right == SIDEMIN) & (Center == CENTERMIN)){
       result = Blocked;
  }
  else if((Left == SIDEMIN) & (Right == SIDEMIN) & (Center == CENTEROPEN)) {
                result = Straight;
    }

  else if((Left == SIDEMAX) & (Right == SIDEMIN) & (Center == CENTERMIN)) {
                result = LeftTurn;
    }
  else if((Left == SIDEMIN) & (Right == SIDEMAX) & (Center == CENTERMIN)) {
                result = RightTurn;
    }
  else if((Left == SIDEMIN) & (Right == SIDEMAX) & (Center == CENTEROPEN)) {
                result = RightJoint;
    }
  else if((Left == SIDEMAX) & (Right == SIDEMIN) & (Center == CENTEROPEN)) {
                result = LeftJoint;
    }
  else if((Left == SIDEMAX) & (Right == SIDEMAX) & (Center == CENTEROPEN)) {
                result = CrossRoad;
    }
  else if((Left == SIDEMAX) & (Right == SIDEMAX) & (Center == CENTERMIN)) {
                result = TeeJoint;
    }
  else if(Left < SIDEMIN) {
      result = LeftTooClose;
  }
  else if(Right < SIDEMIN) {
    result = RightTooClose;
  }
  else if(Center < CENTERMIN) {
    result = CenterTooClose;
  }
  else if((Left < 50) || (Right < 50) || (Center < 50) || (Left > 800) || (Right > 800)
  || (Center > 800)) {
                 result = Error;
     }
  else{
      result = Error;
  }

  return result;
}

#define IRSlope 1195172
#define IROffset -1058
#define IRMax 2552
#define DMax 800

int32_t Convert(int32_t n){
    int32_t D = (IRSlope)/(n + IROffset);//convert here

  return D;
}
// **********testing of Convert*********
int32_t const ADCBuffer[16]={2000, 2733, 3466, 4199, 4932, 5665, 6398, 7131, 7864, 8597,
 9330, 10063, 10796, 11529, 12262, 12995};
int32_t const DistanceBuffer[16]={800, 713, 496, 380, 308, 259, 223, 196, 175, 158, 144,
 132, 122, 114, 106, 100};
void Program4_1(void){
  int i;
  int32_t adc,distance,errors,diff;
  errors = 0;
  for(i=0; i<16; i++){
    adc = ADCBuffer[i];
    distance = Convert(adc); // call to your function
    diff = distance-DistanceBuffer[i];
    if((diff<-1)||(diff>1)){
      errors++;
    }
  }
  while(1){};
}
// ***********end of testing of Convert*********



// ***********testing of classify
scenario_t Solution(int32_t Left, int32_t Center, int32_t Right);
int32_t const CornerCases[18]={49,50,51,149,150,151,211,212,213,353,354,355,599,600,601,799,800,801};
int32_t errors;
void Program4_2(void){
  enum scenario result,truth;
  int i,j,k;
  int32_t left, right, center; // sensor readings
  errors = 0;
  for(i=0; i<18; i++){
    left = CornerCases[i];
    for(j=0; j<18; j++){
      center = CornerCases[j];
      for(k=0; k<18; k++){
        right = CornerCases[k];
        result = Classify(left,center,right); // your solution
        truth = Solution(left,center,right);  // correct answer
        if(result != truth){
           errors++;
        }
      }
    }
  }
  while(1){
  }
}

void Program4_3(void){ // will take over 16 hours to complete
  enum scenario result,truth;
  int32_t left, right, center; // sensor readings
  errors = 0;
  for(left=0; left<1000; left++){
    for(center=0; center<1000;  center++){
      for(right=0; right<1000; right++){
        result = Classify(left,center,right); // your solution
        truth = Solution(left,center,right);  // correct answer
        if(result != truth){
           errors++;
        }
      }
    }
  }
  while(1){
  }
}

void main(void){
  // run one of these
 Program4_1();
 //Program4_2();
//  Program4_3();
}
```