

```

// Spectrometer_led10

// clear the LCD
void clearLCD() {
    Serial.print(12, BYTE);
    delay(1);
}

// start a new line
void newLine() {
    Serial.print(10, BYTE);
}

// variable declaration
int inPinRed = 3; // red button on pin 3
int inPinBlack = 2; // black button on pin 2
int redState; // is red button pressed
int blackState; // is black button pressed
int val = 0;
int analogPin = 1; // sensor output connected to Analog Pin 1
int sensorValue = 0; // variable to store the ADC value of the photodiode
int red = 6; // red led on output pin 6
int yellow = 7; // yellow led on output pin 7
int ir = 8; // IR led on output pin 8
int blue = 9; // blue led on output pin 9
int orange = 10; // orange led on output pin 10
int pinArray[] = {6, 7, 8, 9, 10}; // pin address array
int count = 0; // for next loop counter
int sampleValue[5]; // sample value array

int windex[] = {38,1019,564,321,315}; // database values for windex
int windexValue; // windex value
int windexTemp = 0; // windex value - sensor value for ID

int motoroil[] = {31,1020,787,283,193};
int motoroilValue;
int motoroilTemp = 0;

int coffee[] = {38,993,41,351,286};
int coffeeValue;
int coffeeTemp = 0;

int softsoap[] = {31,1020,708,379,58};
int softsoapValue;
int softsoapTemp = 0;

int water[] = {33,1017,582,327,45};

```

```

int waterValue;
int waterTemp = 0;

int mouthwash[] = {46,1020,621,292,568};
int mouthwashValue;
int mouthwashTemp = 0;

int tea[] = {33,1010,335,317,48};
int teaValue;
int teaTemp = 0;

int rainx[] = {32,1015,402,305,49};
int rainxValue;
int rainxTemp = 0;

int nosample[] = {46,488,30,44,325};
int nosampleValue;
int nosampleTemp = 0;

int none[] = {46,488,30,44,325}; // no sample
int noneValue;
int noneTemp = 0;

void setup() // initialize settings
{
  beginSerial(19200); //set serial port to 19.2k baud
  pinMode(inPinRed, INPUT); // set red switch pin as input
  pinMode(inPinBlack, INPUT); // set black switch pin as input
  pinMode(red, OUTPUT); // set red led pin as output
  pinMode(yellow, OUTPUT); // set yellow led pin as output
  pinMode(ir, OUTPUT); // set IR led pin as output
  pinMode(blue, OUTPUT); // set blue led pin as output
  pinMode(orange, OUTPUT); // set orange led pin as output

  // actually set the pins for output
  for (count=0;count<4;count++) {
    pinMode(pinArray[count], OUTPUT);
  }

  // Print startup text
  delay(100);
  // nex long line is for initializing the display when necessary
  //Serial.print(26,BYTE), Serial.print(26,BYTE), Serial.print(32,BYTE),Serial.print
  (32,BYTE),Serial.print(32,BYTE),Serial.print(32,BYTE),Serial.print
  (32,BYTE),Serial.print(32,BYTE),Serial.print(26,BYTE),Serial.print(26,BYTE); //
  reboot display

```

```
//Serial.print(9,BYTE), Serial.print(3,BYTE); // Splash screen called from Display  
EEPROM  
Serial.print(15,BYTE), Serial.print(60,BYTE); //Display Contrast  
Serial.print(23,BYTE); //Wrap on  
Serial.print(20,BYTE); //Scroll off  
Serial.print(4,BYTE); //hide cursor  
//Serial.print(10, BYTE), Serial.print(13,BYTE); //line feed, carriage return
```

```
clearLCD();
```

```
Serial.print("  Liquid ID  ");  
Serial.print("  Spectrometer  ");  
Serial.print("  System  ");  
Serial.print("Rev 1.10  June 2007");
```

```
//Serial.print(9,BYTE), Serial.print(5,BYTE); //Uncomment to send CTech splash  
screen to display EEPROM
```

```
delay(3000); // keep the welcome screen on for 3 seconds
```

```
digitalWrite(pinArray[0], LOW); // turn off the red led
```

```
clearLCD();  
}
```

```
// program starts here
```

```
void loop() {
```

```
    clearLCD();
```

```
    Serial.print("  Transmission Mode ");  
    Serial.print("                ");  
    Serial.print("                ");  
    Serial.print("  Reflection Mode  ");  
    delay(50);
```

```
    for (;;) { // wait for button press  
        redState = digitalRead(inPinRed); // test for red button press  
        blackState = digitalRead(inPinBlack); // test for black button press  
        if (redState == LOW) // red button pressed  
        {  
            goto transmission; // jump to transmission subroutine  
        }  
    }
```

```

    if (blackState == LOW) // black button pressed
    {
        goto reflection; // jump to reflection subroutine
    }
}

```

transmission: // transmission subroutine

```

    clearLCD();
    Serial.print(" Transmission Mode");
    delay(500);
    clearLCD();
    Serial.print(" Identify      ");
    Serial.print(" ");
    Serial.print(" ");
    Serial.print(" Learn      ");
    delay(100);
    for (;;) { // wait for button press
        redState = digitalRead(inPinRed); // test for red button press
        blackState = digitalRead(inPinBlack); // test for black button press
        if (redState == LOW) // red button pressed
        {
            goto identify; // jump to identify subroutine
        }
        if (blackState == LOW) { // black button pressed
            goto learn; // jump to learn subroutine
        }
    }
}

```

learn: // learn subroutine

```

    clearLCD();
    Serial.print(" Learn Mode ");
    for (count=0;count<5;count++) {
        digitalWrite(pinArray[count], HIGH); //turn the led on
        delay(100); // wait for stabilization
        sensorValue = analogRead(analogPin); // read the value from the sensor
        if (sensorValue <= 99) // test for number of digits to display
        {Serial.print(32, BYTE); // insert space
        }
        if (sensorValue <= 9) // test for number of digits to display
        {Serial.print(32, BYTE); // insert space
        }
        Serial.print(sensorValue),Serial.print("-"); //send as a ascii encoded number
        delay(30);
        if (count == 0)
        {Serial.print("Red ");

```

```

}
if (count == 1)
{Serial.print("Ylw  ");
}
if (count == 2)
{Serial.print("IR  ");
}
if (count == 3)
{Serial.print("Blue  ");
}
if (count == 4)
{Serial.print("Ornge "),Serial.print(1,BYTE);
}

```

```

delay(20);

```

```

digitalWrite(pinArray[count], LOW); // turn led off
delay(100);
}

```

```

for (;;) { // wait for button press
  redState = digitalRead(inPinRed); // test for red button press
  blackState = digitalRead(inPinBlack); // test for black button press
  if (redState == LOW){goto transmission;} // on red button press jump to
transmission subroutine
  if (blackState == LOW){goto reflection;} // on black button press jump to reflection
subroutine
}

```

```

reflection: // reflection subroutine

```

```

clearLCD();
Serial.print("          ");
Serial.print("          ");
Serial.print("          ");
Serial.print(" Reflection Mode ");
delay(500);

```

```

// reflection mode code here

```

```

for (;;) { // wait for button press
  redState = digitalRead(inPinRed); // test for red button press
  blackState = digitalRead(inPinBlack); // test for black button press
  if (redState == LOW){goto transmission;} // on red button press jump to
transmission subroutine
  if (blackState == LOW){goto reflection;} // on black button press jump to reflection

```

```
subroutine
}
```

```
identify: // identify subroutine
```

```
clearLCD();
```

```
Serial.print(" Identify Mode ");
Serial.print(" ");
Serial.print(" ");
Serial.print(" ");
delay(500);
clearLCD();
```

```
// Identify mode code
windexTemp=0; // reinitialize
softsoapTemp=0;
coffeeTemp=0;
motoroilTemp=0;
waterTemp=0;
mouthwashTemp=0;
teaTemp=0;
rainxTemp=0;
nosampleTemp=0;
noneTemp=0;
```

```
for (count=0;count<5;count++) { //load the matrix with five color values
digitalWrite(pinArray[count], HIGH); //turn on led
delay(100); // wait for led to stabilize
sensorValue = analogRead(analogPin); // read the value from the sensor
sampleValue[count] = sensorValue; // store value in matrix
digitalWrite(pinArray[count], LOW); // turn off led
```

```
windexValue= abs((sampleValue[count])-(windex[count])); // subtract database
value from sampled value for each color
windexTemp=((windexValue)+(windexTemp)); // add up values from matrix for
windex
windexValue=0;
```

```
softsoapValue= abs((sampleValue[count])-(softsoap[count]));
softsoapTemp=((softsoapValue)+(softsoapTemp));
softsoapValue=0;
```

```
coffeeValue= abs((sampleValue[count])-(coffee[count]));
coffeeTemp=((coffeeValue)+(coffeeTemp));
coffeeValue=0;
```

```
motoroilValue= abs((sampleValue[count])-(motoroil[count]));
motoroilTemp=((motoroilValue)+(motoroilTemp));
motoroilValue=0;
```

```
waterValue= abs((sampleValue[count])-(water[count]));
waterTemp=((waterValue)+(waterTemp));
waterValue=0;
```

```
mouthwashValue= abs((sampleValue[count])-(mouthwash[count]));
mouthwashTemp=((mouthwashValue)+(mouthwashTemp));
mouthwashValue=0;
```

```
teaValue= abs((sampleValue[count])-(tea[count]));
teaTemp=((teaValue)+(teaTemp));
teaValue=0;
```

```
rainxValue= abs((sampleValue[count])-(rainx[count]));
rainxTemp=((rainxValue)+(rainxTemp));
rainxValue=0;
```

```
nosampleValue= abs((sampleValue[count])-(nosample[count]));
nosampleTemp=((nosampleValue)+(nosampleTemp));
nosampleValue=0;
```

```
noneValue= abs((sampleValue[count])-(none[count]));
noneTemp=((noneValue)+(noneTemp));
noneValue=0;
```

```
}
```

```
for (count=0;count<250;count=count++) // test for minimum value (needs to be
rewritten with min max)
```

```
{
  if (windexTemp <= count)
  {
```

```
    clearLCD();
    Serial.print("  Windex ");
    break;
  }
```

```
  if (softsoapTemp <= count)
  {
    clearLCD();
    Serial.print("  Soft Soap ");
    break;
  }
```

```

}

if (motoroilTemp <= count)
{
  clearLCD();
  Serial.print("  Motor Oil ");
  break;
}

if (coffeeTemp <= count)
{
  clearLCD();
  Serial.print("  Coffee  ");
  break;
}

if (waterTemp <= count)
{
  clearLCD();
  Serial.print("  Water ");
  break;
}
if (mouthwashTemp <= count)
{
  clearLCD();
  Serial.print("  Mouth Wash ");
  break;
}

if (teaTemp <= count)
{
  clearLCD();
  Serial.print("  Tea ");
  break;
}
if (rainxTemp <= count)
{
  clearLCD();
  Serial.print("  Rain-X ");
  break;
}

if (nosampleTemp <= count)
{
  clearLCD();
  Serial.print("  No Sample ");
}

```



```

    break;
}

if (noneTemp <= count)
{
    clearLCD();
    Serial.print("    No Sample ");
    break;
}
    clearLCD();
    Serial.print("    UNKNOWN");
}
for (;;) { // wait for button press
    redState = digitalRead(inPinRed); // test for red button press
    blackState = digitalRead(inPinBlack); // test for black button press
    if (redState == LOW){goto transmission;} // on red button press jump to
transmission subroutine
    if (blackState == LOW){goto reflection;} // on black button press jump to reflection
subroutine
}
}

```