In order to display Celcius, Fahrenheit, and Hue readings on the LCD screen, the highlighted lines of code were added. Sample code was available on the Arduino reference site (http://arduino.cc/en/Reference/LiquidCrystal?from=Tutorial.LCDLibrary) as well as a guide to the different command prompts.

```
#include <LiquidCrystal.h>
#include <i2cmaster.h>
#include "Wire.h"
//#include "BlinkM_funcs.h"

const float lowReading = 60;
const float highReading = 75;
const unsigned char separatorCharacter = 255;
//This line initializes the LCD and designates which pins on the Arduino perform the read/write functions
LiquidCrystal lcd(8, 7, 5, 4, 3, 2);
void setup(){

  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  Serial.begin(9600);
  Serial.println("starting setup...");
  //added these two lines to print the starting setup on LCD
   lcd.begin(16,2);
   lcd.print("Starting setup...");

   delay (2000);

  i2c_init(); //Initialise the i2c bus
  PORTC = (1 << PORTC4) | (1 << PORTC5);//enable pullups
  Serial.println("completed setup");
  //clears previous text and writes next line
  lcd.clear();
  lcd.print("Completed Setup");
  //tells arduino to wait 2 seconds before clearing the screen
  delay (2000);
  lcd.clear();
}

float normf(float x, float low, float high) {
  float y = (x - low) * 255.f / (high - low);
  if(y > 255) {
    y = 255;
  }
  if(y < 0) {
    y = 0;
  }
  return y;
```

```
}

void loop(){
  int dev = 0x5A<<1;
  int data_low = 0;
  int data_high = 0;
  int pec = 0;

  i2c_start_wait(dev+I2C_WRITE);
  i2c_write(0x07);

  // read
  i2c_rep_start(dev+I2C_READ);
  data_low = i2c_readAck(); //Read 1 byte and then send ack
  data_high = i2c_readAck(); //Read 1 byte and then send ack
  pec = i2c_readNak();
  i2c_stop();

  //This converts high and low bytes together and processes temperature, MSB is a error bit and is
ignored for temps
  double tempFactor = 0.02; // 0.02 degrees per LSB (measurement resolution of the MLX90614)
  double tempData = 0x0000; // zero out the data
  int frac; // data past the decimal point

  // This masks off the error bit of the high byte, then moves it left 8 bits and adds the low byte.
  tempData = (double)(((data_high & 0x007F) << 8) + data_low);
  tempData = (tempData * tempFactor)-0.01;
  //This is where you calibrate the sensor by adjusting the value in the kalvin to celcius conversion
formula (274.15)
  float celcius = tempData - 274.15;
  float fahrenheit = (celcius*1.8) + 32;

  //Serial.println(fahrenheit);

  float state = normf(fahrenheit, lowReading, highReading);
  //Serial.write((unsigned int) state);
  //Serial.write(separatorCharacter);

  // BlinkM MaxM super-bright LED:
  // 165 is blue, 0 is red
  //BlinkM_fadeToHSB(blinkm_addr, map(state, 0, 255, 165, 0), 255, 255);

  // Regular ol' RGB LED:
  int hue = map(state,0,255,359,(359*0.5)); // not the whole color wheel
  setLedColorHSV(hue,1,1); //We are using Saturation and Value constant at 1

  Serial.print(fahrenheit);
  Serial.print(" degrees F, hue: ");
  Serial.println(+hue);
```

```
  // these next lines are the  body of the code for the LCD screen
  //clear lcd screen
  lcd.clear();
  //move cursor
  lcd.setCursor(0,0);
  //print variable "farenheit" followed by custom character for degrees followed by "F"
  lcd.print(fahrenheit);
  lcd.print((char)223);
  lcd.print("F ");
  //print variable "celcius" followed by custom character for degrees followed by "C"
  lcd.print(celcius);
  lcd.print((char)223);
  lcd.print("C");
  //move cursor to next line down
  lcd.setCursor(0,2);
  // print variable "+Hue"
  lcd.print("Hue: ");
  //
  lcd.print(+hue);
  //Arduino waits .3 seconds before repeating loop
  delay (300);




}
//Convert a given HSV (Hue Saturation Value) to RGB(Red Green Blue) and set the led to the color
//  h is hue value, integer between 0 and 360
//  s is saturation value, double between 0 and 1
//  v is value, double between 0 and 1
//http://splinter.com.au/blog/?p=29
void setLedColorHSV(int h, double s, double v) {
  //this is the algorithm to convert from RGB to HSV
  double r=0;
  double g=0;
  double b=0;

  double hf=h/60.0;

  int i=(int)floor(h/60.0);
  double f = h/60.0 - i;
  double pv = v * (1 - s);
  double qv = v * (1 - s*f);
  double tv = v * (1 - s * (1 - f));

  switch (i)
  {
  case 0: //rojo dominante
    r = v;
```

```
    g = tv;
    b = pv;
    break;
  case 1: //verde
    r = qv;
    g = v;
    b = pv;
    break;
  case 2:
    r = pv;
    g = v;
    b = tv;
    break;
  case 3: //azul
    r = pv;
    g = qv;
    b = v;
    break;
  case 4:
    r = tv;
    g = pv;
    b = v;
    break;
  case 5: //rojo
    r = v;
    g = pv;
    b = qv;
    break;
  }

  //set each component to a integer value between 0 and 255
  int red=constrain((int)255*r,0,255);
  int green=constrain((int)255*g,0,255);
  int blue=constrain((int)255*b,0,255);

  setLedColor(red,green,blue);
}

//Sets the current color for the RGB LED
void setLedColor(int red, int green, int blue) {
  //Note that we are reducing 1/4 the intensity for the green and blue components because
  //  the red one is too dim on my LED. You may want to adjust that.
  analogWrite(9,red); //Red pin attached to 9
  analogWrite(10,green); //Red pin attached to 9
  analogWrite(11,blue); //Red pin attached to 9
}
```