Lux Sensor and Digital Display

Patrick M. Munsey, Kyle D. Waas

Dordt University, EGR 304

March 24th, 2020

Incorrect 1st-page heading.This weakens the impact of

I. Introduction

This report explores the power supply breadboarding and the void loop and float collection coding techniques used to create a lux sensor to display interface. The BH1750 lux sensor and 0.96” I2C digital display were constructed prior to this project and its designs and construction will not be covered in this report. An arduino system, breadboard, and accompanying wiring was also provided by Dordt University to complete the projects needed materials.

II. Method

This project was greatly assisted by an Instructables project by nafisaanjum13 [1]. The link is included in the references and noted here since this page will be mentioned frequently throughout this section.

For our power supply breadboarding, the specifications and limitations of both instruments were taken into consideration. Since the BH1750 and digital display weren’t both rated similarly in voltage power, they needed to be connected to different nodes coming out of

What is "voltage power?" What units would that be?

the Arduino system. The digital display was the larger of the two and worked well on the stable five volt output pin that the Arduino comes ~~programmed~~ with. The sensor was also just as easy

A power pin is not "programmed."

to power up since Arduinos also come with a built in 3v3 pin that adheres well to the BH1750’s

"3v3" is a European slang code notation properly used only on small parts and

2.4-3.6 voltage needs.

silk-screenings where space is limited. (Like a color code.) You mean "3.3 V" which

This deviated from the instructions put forth by Instructables as it ~~was~~ suggested to ~~only~~

is the way the Arduino Uno board is silk-screened. ("3v3" is an informal extension of

use the 3v3 pin to power both instruments. However, no complications were found through our

a legitimate code for labeling resistors and capacitors, the IEC 60062:2016 code.)

methodology and it was decided to not fix a working design(Figure 1).

You imply that your circuit does not exactly match what is given in the Instructable. Th

For the interface programming, the Arduino IDE program was used to communicate with the arduino system and subsequently, the instrument(Figure 2,3,4). The set up of the program

A space is needed.

proved to be the most intricate process we encountered. The Instructable page provided ~~a~~ an .ino file with a working program, however, the digital display did not seem to sync up within the program's set up. To troubleshoot this, we used a test program that searched for the I2C port connection and gave feedback on the connection status(Figure 5&6). Through this we found the correct communication pins and proceeded with the given .ino file. Libraries needed were also provided on the Instructables page[1].

The second half of the program is for the communication aspect between the sensor and display. A ~~void~~ loop was used as the basis of the Arduino's instructions. A ~~void~~ loop in the

That the loop is of the type, "void" is so unimportant in this context that your writing losses authority when you mention it. See for one example the Wikipedia article that explains what the type "void" means: https://en.wikipedia.org/wiki/Void_type

You have not defined what a "Maker Board" is. The phrase also does not appear anywhere in your reference--the Instructables page.

Arduino IDE instructs the system and its attached instruments to continue operating for as long as the Maker Board is active. Within this loop, our incoming lux value from the BH1750 is accepted at a 'float' in order to display a couple decimal places of precision that the sensor can give us. This is important since leaving this out would under-utilize our purchased equipment. ✓ Good We acknowledged that in circumstances where the two decimal precision isn't valuable to the user, a cheaper sensor could be purchased and the 'float' left out in order to keep a program cleaner and thinner. This thinning would then allow our system to run faster with less CPU usage. Had we been building a protective device for someone's eyes that needed to read the lux value of the user's room and polarize a set of lenses in response to higher values, a three to four millisecond delay could not be as acceptable. However, with our set up and program being very rudimentary, we decided to keep the feature and show as much precision as we were allowed.

III. Results

Though unsuccessful at first due to a miswired I2C connection at the digital display, we did get a functioning light sensor that would read out its current lux intake through a digital 128x64 pixel display. I guess I'll just have to take your word for it--and the demo. However you could have performed some sort of testing, even just covering the sensor with yourhand and re

IV. Conclusion

The project was successful and created a light sensor that fit our needs. The decision to keep the precision aspect of our project was deemed suitable since there was not a time related aspect to our needs. Because the device was created as a hobby piece and a skill building exercise in I2C communication buses, the delay in sensor readout is without consequence.

Progress seen: In the lab--you get things working and have more confidence now. On pap

References

[1] nafisaanjum13. “Mini Digital LUX Meter.” Instructables.

https://www.instructables.com/id/Mini-Digital-LUX-Meter/. (accessed Feb. 12th, 2020)

Figures

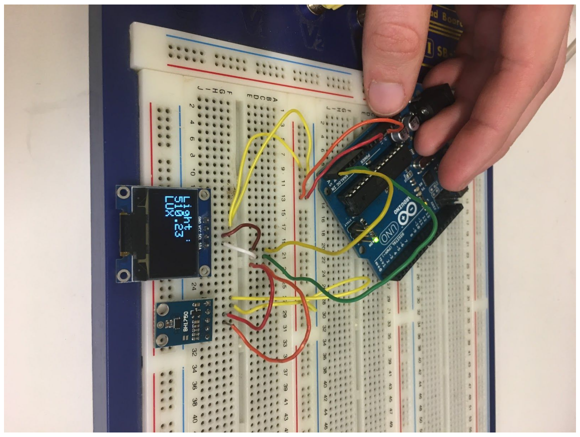
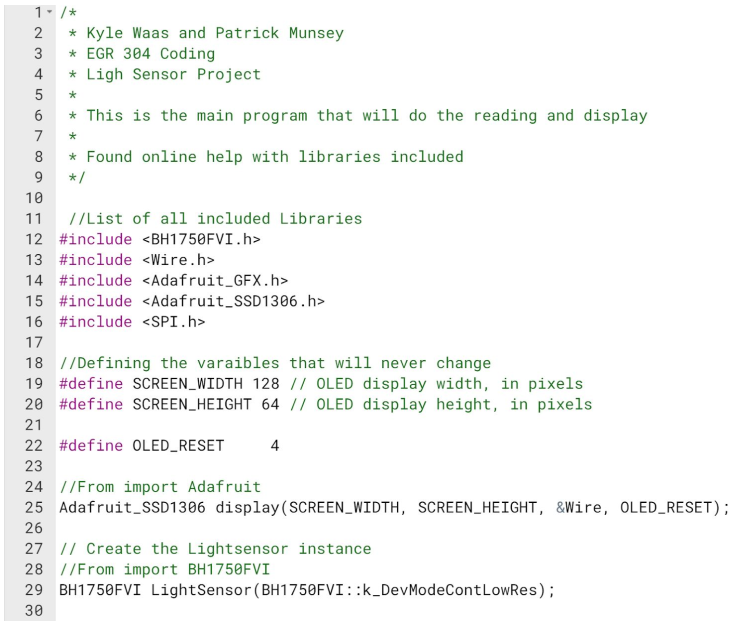
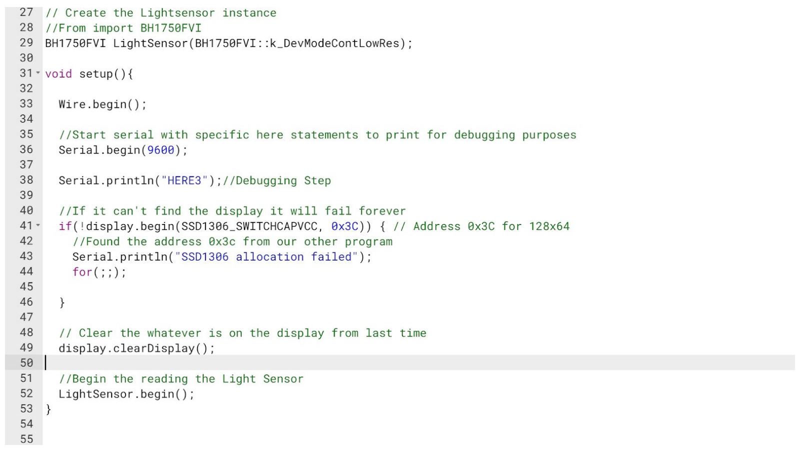


Fig. 1, Photo of the project as built.
All figures need captions, even if the subject of the figure seems obvious.



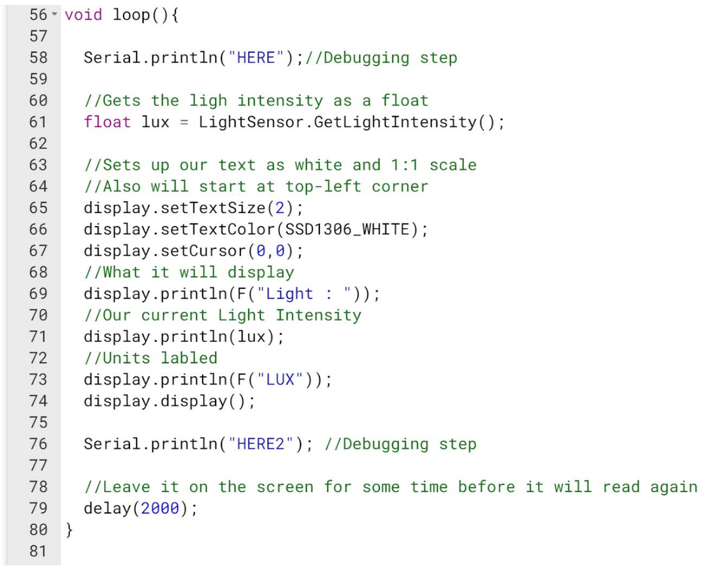
```
/*
 * Kyle Waas and Patrick Munsey
 * EGR 304 Coding
 * Ligh Sensor Project
 *
 * This is the main program that will do the reading and display
 *
 * Found online help with libraries included
 */

 //List of all included Libraries
#include <BH1750FVI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SPI.h>

//Defining the varaibles that will never change
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

#define OLED_RESET     4

//From import Adafruit
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Create the Lightsensor instance
//From import BH1750FVI
BH1750FVI LightSensor(BH1750FVI::k_DevModeContLowRes);

```

Fig. 2 Missing caption. . . same for all following figures.



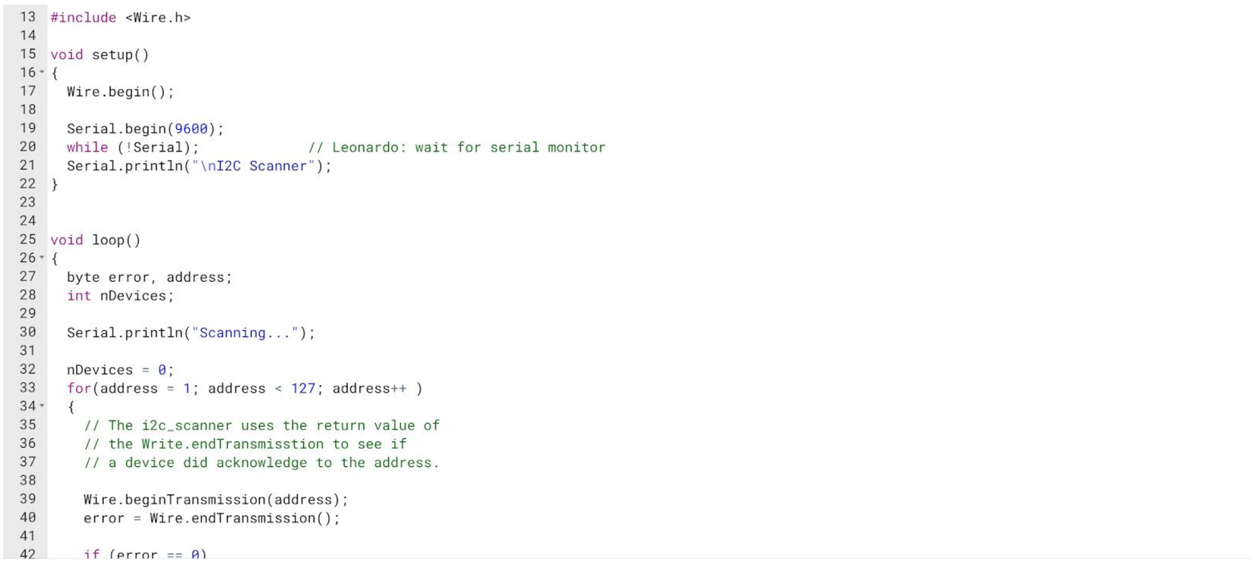
```
// Create the Lightsensor instance
//From import BH1750FVI
BH1750FVI LightSensor(BH1750FVI::k_DevModeContLowRes);

void setup(){

  Wire.begin();

  //Start serial with specific here statements to print for debugging purposes
  Serial.begin(9600);

  Serial.println("HERE3");//Debugging Step

  //If it can't find the display it will fail forever
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x64
    //Found the address 0x3c from our other program
    Serial.println("SSD1306 allocation failed");
    for(;;);

  }

  // Clear the whatever is on the display from last time
  display.clearDisplay();

  //Begin the reading the Light Sensor
  LightSensor.begin();
}


```

Fig. 3



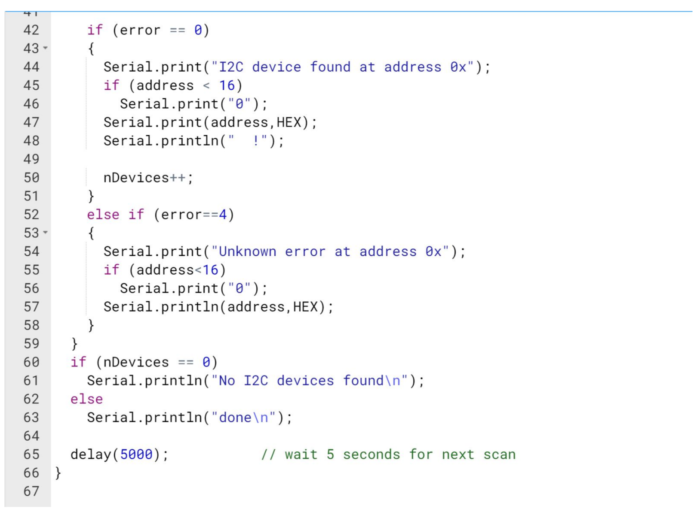
```
void loop(){

  Serial.println("HERE");//Debugging step

  //Gets the ligh intensity as a float
  float lux = LightSensor.GetLightIntensity();

  //Sets up our text as white and 1:1 scale
  //Also will start at top-left corner
  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  //What it will display
  display.println(F("Light : "));
  //Our current Light Intensity
  display.println(lux);
  //Units labled
  display.println(F("LUX"));
  display.display();

  Serial.println("HERE2"); //Debugging step

  //Leave it on the screen for some time before it will read again
  delay(2000);
}

```

Fig. 4



```
#include <Wire.h>

void setup()
{
  Wire.begin();

  Serial.begin(9600);
  while (!Serial);             // Leonardo: wait for serial monitor
  Serial.println("\nI2C Scanner");
}


void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.

    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
```

Fig. 5



```
    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknown error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);           // wait 5 seconds for next scan
}
```

Fig. 6

__B-__Style: Most of the comments above have to do with style.__C+__Completeness: