CMPE 110 – Electronics for Computing Systems

Professor Cuong Nguyen Section 4 Monday/Wednesday 6:00-6:50 PM

**Bluetooth-Enabled Optical Sensor**

**using ADC**

*Group 15*

*Thinh Le(010641462)1, Troy Kurniawan(010543013)2, and Varinder Singh(009620871)3*

***Abstract* - This report discusses a project that involved creating a bluetooth-controlled night light. The structure of our project consists of an Arduino Uno R3 board, which contains an ATMEGA328P-PU microcontroller, to communicate across several components: a bluetooth device, optical light sensor, temperature sensor, ultrasonic sensor and voice recognition. Once the bluetooth activates the board, the buzzer makes a beeping sound. When exposed to very little light, the machine’s LED lights up. This machine effectively functions as a bluetooth activated night light.**

**Introduction:**

The project designed and built a bluetooth-enabled optic-sensing LED and buzzer interfaced with a serial seven-segment display through an SPI. The main purpose of the project was to test our understanding of electronics to create a machine useful for everyday application. For this project, a bluetooth-controlled night light was built. This machine would be a convenient alternative to people who want a night light. Each member of the group project has a different level of experience. Nonetheless, each member studied more about the topic to obtain the knowledge needed to build the machine. The highlight of this project was the bluetooth implementation. Bluetooth added the convenience of turning the nightlight on and off. One of the main issues of the project was learning new concepts and ideas to have all of the components working as a single machine, especially when it came to the SPI and the bluetooth features. Another concern was time management. Time was admittedly not used effectively. In the end, the project worked successfully.

**Project Specs:**

The main goal of this project was to create a simple bluetooth controlled night light. It was made with the intention that anyone can turn their night light on and off using a mobile device. Other goals included understanding the various electrical components. Understanding how these parts function will certain help for future reference.

**Project Justification:**

The project idea was first created by gathering several different components that would act as a sensor, a control signal, and/or an output. After much discussion, certain components were selected to be implemented for the final product: a light sensor, a bluetooth control signal, a MOSFET, an LED, a buzzer, and a seven-segment display. This is how the bluetooth-controlled night light was brought about.

*MOSFET and Photoresistor*

The plan was to control the brightness of the LED using the photoresistor. In order to do that, a circuit that included the power supply, a 100kΩ resistor, a photoresistor, and an N-channel MOSFET was constructed. The MOSFET is essentially a voltage-controlled switch. The idea was to use the photoresistor to adjust the voltage that goes into the “G” input of the MOSFET. The MOSFET then would control the LED by blocking or not blocking the current that flows through its “D” and “S” pins. In addition, according to the datasheet, the MOSFET’s “G” pin needs anywhere between 1V and 2.5V to enter saturation mode and close its switch from the “D” pin to the “S” pin. The photoresistor controls the voltage going into the MOSFET by acting as a pull-down resistor. In the circuit, there is a 5V power supply connected in series with a 100kΩ resistor. At the end of this resistor is the MOSFET and the photoresistor (which is connected to ground). Using Kirchhoff’s Voltage Laws, the effects of the photoresistor on the MOSFET can be seen.

When the photoresistor is exposed to light, its resistance is roughly 4kΩ.

When the photoresistor is blocked from light, its resistance is roughly 100kΩ.

Therefore, when the photoresistor is exposed to light, 0.19V goes into the MOSFET’s “G” pin, which is not enough to close the switch, letting the LED stay dormant. However, when the photoresistor is exposed to light, 2.5V goes into the MOSFET’s “G” pin, which is enough to close the switch and turn on the LED.

*Buzzer*

The buzzer was used to signify when the arduino board for the LED turned on and off. When the bluetooth is turned on, the buzzer beeps twice. When the bluetooth is turned off, the buzzer beeps once. There are two separate code chunks that describe how the buzzer beeps. When the bluetooth turns on, the code tells the digital GPIO pin to output a high and low signal with delay in a way that causes the buzzer to ring twice. When the bluetooth turns off, the code tells the digital GPIO pin to output a high and low signal with delay in a way that causes the buzzer to ring once.

*Ohm-Meter*

Considering we are designing an optic sensor, we needed a method to deduce the ambient level of the light. Since the photoresistor is light-sensitive and changes resistor value based on the presence of light, we could use ohms (Ω) to measure ambience. An ohm-meter circuit was designed to use voltage division and the Uno’s Analog inputs to calculate the value of the photoresistor as light changes (ADC). We used a known resistor value (100kΩ) to calculate the photoresistor in the Arduino IDE, observed in **Figure 3**.

The Vcc is 5V and the resolution is 10 bits (0-1023), thus, the sampling rate is 4.9mV/bit.

Before we could measure the photoresistor, we needed to have a test circuit of two known resistor values. We had the following 4 trials:

|  |
| --- |
| 1st Time  R1: 19.71kΩ, R2: 29.8kΩ  2nd time (IC)  R1: 10.63kΩ, R2: 12.34kΩ  3rd time  R1: 12.1kΩ, R2: 12.4kΩ  4th time  R1:12.11kΩ, R2: 12.43kΩ |

We proceeded to construct our ohmmeter circuit as seen in **Figure 7.** Afterwards, we compiled the arduino code and received the following outputs for R2: (1) 7069.38Ω and (2) 8014.29Ω. Then we calculated the percent error and went with the lower error (33.76%).

|  |
| --- |
| Percent Error for the Ohm Detector   1. = |

*Serial 7-Segment Display*

Now that we had the ambient level of the light, we needed a way to display that information in order to observe the ambience. The Serial 7-Segment (S7S) Display is used in our project as an interface to the user to analyze the ambient level of the room.

Based off of the SH5461AS Datasheet, the S7S was drawn in **Figure 6** to portray the 12 pins of the component. As noticed, there are four pins (6, 8, 9, and 12) that are used to ground the circuit. They represent the 4 digits on the display meanwhile the 7 other pins (pin 3 is excluded) are used to send voltage through the 7 diodes that light up the segmented display. A visual representation of this can be seen in the comments at the top of **Figure 3.**

Before wiring the S7S, we needed to know a safe resistor value to use on the circuit to avoid shorting out the display. Referring to the datasheet and ohm’s law, we had the following results:

|  |
| --- |
| (used 1k)  Duty Cycle: 10%  Frequency: 10kHz  Period: 10f |

*Serial Peripheral Interface*

Since the project consisted of two Arduino Uno boards, we needed a method to communicate values across the boards. Our interface between the two boards existed on the breadboard where the *blue Arduino* supplied DC voltage to the photoresistor/diodes while the *black Arduino* measured the sampling rate of the resistors through the analog pins. The baud rate, or data exchange rate, was 9600.

*Bluetooth*

After the both Arduino’s are powered on, the bluetooth device is enabled using a simple LED application which activate and deactivates pin 13 supplying voltage into the board and the buzzer. The bluetooth device receives power from the Arduino and the RX and TX outputs on the Arduino are assigned the TX and RX inputs on the Arduino respectively. This allows the Arduino to receive instructions from the bluetooth device.

After this point, the 1kΩ resistor is supplied with voltage while the 100kΩ resistor recives power from the Genuino. The photoresistor is placed in parallel with the 100kΩ resisotor and the drain pin. When the photoresistor is supplied with light, the LED remain off because the voltage flowing into the gate isn’t enough to reach the saturation mode. However, when the photoresistor is covered the, the resistance of the photoresistor increases and the voltage going into the gate rises to about 4.7V, reaching saturation mode and allowing current to flow into the LED. The LED applications on and off buttons are also synchronized with active buzzer. The arduino pin 13 supplies voltage to the buzzer and this supply controlled in order to indicate when the voltage is supplied to the board and when it is turned off.

Schematics can be found in the appendix section of the report.

*Parts List:*

* Arduino Uno
* Arduino Genuino
* Active Buzzer
* LED white
* 100kΩ Resistor
* 1kΩ Resistor
* Photoresistor
* 7805CT N-channel MOSFET
* 4 Digit 7-Segment Display (SH5461AS)

# DSD TECH HC-05 Bluetooth

*Tool List:*

* *Arduino*
* *LTSpice*
* *LogicWorks*

**Contribution:**

|  |  |  |
| --- | --- | --- |
| **THINH LE:**   * Designed the seven-segment display and serial peripheral interface (SPI) to display the photoresistor intensity (ohmmeter). * Wrote code for the seven-segment display, ohmmeter, and SPI. | **TROY KURNIAWAN:**   * Designed part of the LED/Photoresistor/ MOSFET circuit along with Varinder (mostly calculation theories). * Debugged code designed by Varinder and Thinh. * Created an ASM chart along with Varinder. | **VARINDER SINGH:**   * Designed the bluetooth circuit/code that controlled the Arduino Uno board. * Designed part of the LED/Photoresistor/ MOSFET circuit along with Troy (mostly the construction of the circuit). * Created an ASM chart along with Troy. |

**Budget Expenses/Time Spent:**

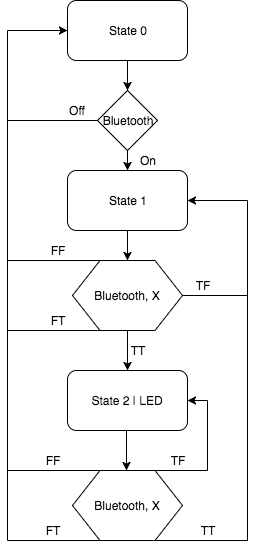
|  |  |
| --- | --- |
| **Budget Expenses:**   * $25 = Thinh’s Electronics Kit (Arduino board, seven-segment display, breadboard, several other components). * $10 = Troy’s Arduino Uno board * $8 = Varinder’s Bluetooth controller * **$43 TOTAL** | **Time Spent:**   * 24 Hours total   + 6 hours for pre-planning   + Day 1: 6 hours for part gathering and coding   + Day 2: 6 hours for implementation and code debugging)   + Day 3: 6 hours for finalization and last minute edits |

**Conclusion:**

The bluetooth-enabled night light project was a success. Although it took quite some time to prepare, design, gather parts, and build, the machine came together in the end. Looking back, the project could have ran more efficiently. For example, time allocation could have been spread out neatly. Last minute adjustments created an inefficient workflow. This was our greatest lesson. Overall, the project was a success.

**Appendix & References:**

*Algorithmic State Machine (ASM) Chart:*

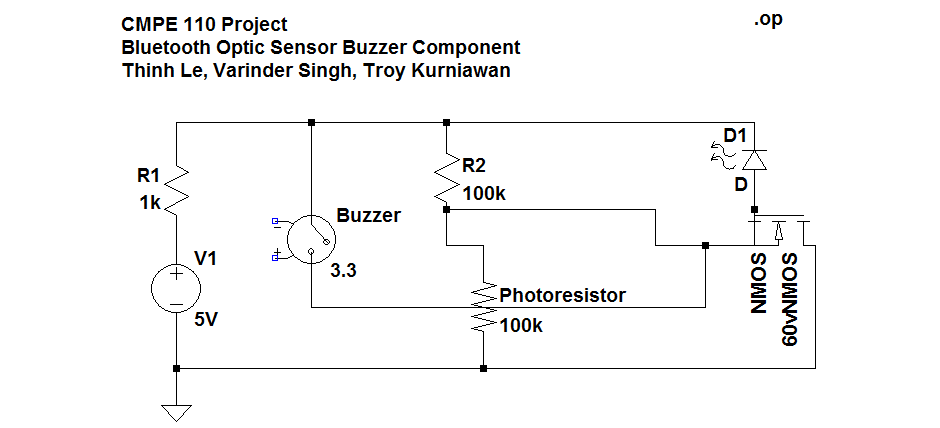


**Figure 1.** ASM Chart

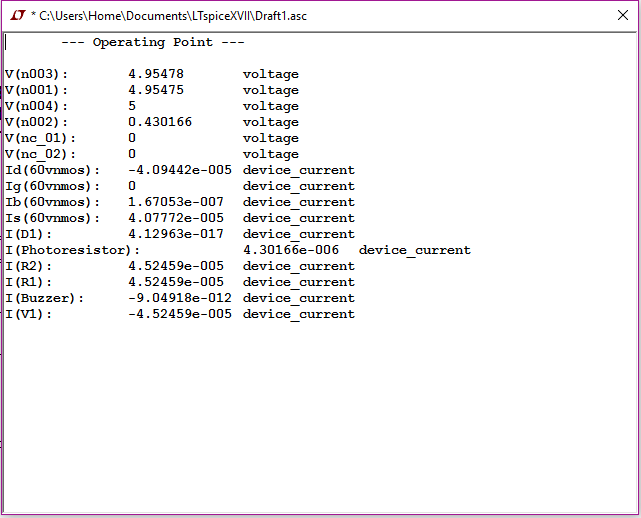
*Source Code:*

|  |
| --- |
| char info = 0;  void setup() {  // put your setup code here, to run once:  Serial.begin(9600); // set the data rate for the serial transmission  pinMode(12,OUTPUT); // pin 11 will output  pinMode(13,OUTPUT); // pin 11 will output  }  void loop() {  // put your main code here, to run repeatedly:  if(Serial.available()>=0 )  {  info = Serial.read();  if(info == '1'){ //"ON" button is pressed  digitalWrite(12, HIGH); //The board is now supplied with +5V    digitalWrite(13, HIGH); // the buzzer actives for 150 ms  delay(50);  digitalWrite(13, LOW);  delay(50);  digitalWrite(13, HIGH);  delay(50);  digitalWrite(13, LOW);    }  else if(info == '0'){ //"OFF" button is pressed  digitalWrite(12, LOW); //The voltage supply to the bread board is 0V  digitalWrite(13, LOW);  digitalWrite(13, HIGH); // the buzzer actives for 50 ms  delay(50);  digitalWrite(13, LOW);  }    }  }  **Figure 2.** Arduino code programing the bluetooth device via the arduino. |
| /\*  Displays digits 0-9 on the 7-segment display  A  ---  F | | B  | G |  ---  E | | C  | |  ---  D  Pins 2-8 connects to the 7 segments of the display  \*/  /\*!< The Ohm Meter \*/  // Represents A0 on UNO board  int analogPin= 1;  // The value from the analog pin A0  int raw= 0;  // The input voltage  int Vin= 5;  // The output voltage  float Vout= 0;  // Resistor value  float R1= 100000; // 100k ohm  float R2= 0;  // Used to calculate R2  float buffer= 0;  /\*!< The Serial 7 Segment Display \*/  int pinA = 2;  int pinB = 3;  int pinC = 4;  int pinD = 5;  int pinE = 6;  int pinF = 7;  int pinG = 8;  int D1 = 9;  int D2 = 10;  int D3 = 11;  int D4 = 13;  // Resistor representation of digits  int digit1 = 0, digit2 = 0, digit3 = 0, digit4 = 0;  void lit(int display){  switch(display){    case 0:  // Set to 0  digitalWrite(pinA, HIGH);  digitalWrite(pinB, HIGH);  digitalWrite(pinC, HIGH);  digitalWrite(pinD, HIGH);  digitalWrite(pinE, HIGH);  digitalWrite(pinF, HIGH);  digitalWrite(pinG, LOW);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    case 1:  // Set to 1  digitalWrite(pinA, LOW);  digitalWrite(pinB, HIGH);  digitalWrite(pinC, HIGH);  digitalWrite(pinD, LOW);  digitalWrite(pinE, LOW);  digitalWrite(pinF, LOW);  digitalWrite(pinG, LOW);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    case 2:  // Set to 2  digitalWrite(pinA, HIGH);  digitalWrite(pinB, HIGH);  digitalWrite(pinC, LOW);  digitalWrite(pinD, HIGH);  digitalWrite(pinE, HIGH);  digitalWrite(pinF, LOW);  digitalWrite(pinG, HIGH);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    case 3:  // Set to 3  digitalWrite(pinA, HIGH);  digitalWrite(pinB, HIGH);  digitalWrite(pinC, HIGH);  digitalWrite(pinD, HIGH);  digitalWrite(pinE, LOW);  digitalWrite(pinF, LOW);  digitalWrite(pinG, HIGH);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    case 4:  // Set to 4  digitalWrite(pinA, LOW);  digitalWrite(pinB, HIGH);  digitalWrite(pinC, HIGH);  digitalWrite(pinD, LOW);  digitalWrite(pinE, LOW);  digitalWrite(pinF, HIGH);  digitalWrite(pinG, HIGH);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    case 5:  // Set to 5  digitalWrite(pinA, HIGH);  digitalWrite(pinB, LOW);  digitalWrite(pinC, HIGH);  digitalWrite(pinD, HIGH);  digitalWrite(pinE, LOW);  digitalWrite(pinF, HIGH);  digitalWrite(pinG, HIGH);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    case 6:  // Set to 6  digitalWrite(pinA, LOW);  digitalWrite(pinB, LOW);  digitalWrite(pinC, HIGH);  digitalWrite(pinD, HIGH);  digitalWrite(pinE, HIGH);  digitalWrite(pinF, HIGH);  digitalWrite(pinG, HIGH);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    case 7:  // Set to 7  digitalWrite(pinA, HIGH);  digitalWrite(pinB, HIGH);  digitalWrite(pinC, HIGH);  digitalWrite(pinD, LOW);  digitalWrite(pinE, LOW);  digitalWrite(pinF, LOW);  digitalWrite(pinG, LOW);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    case 8:  // Set to 8  digitalWrite(pinA, HIGH);  digitalWrite(pinB, HIGH);  digitalWrite(pinC, HIGH);  digitalWrite(pinD, HIGH);  digitalWrite(pinE, HIGH);  digitalWrite(pinF, HIGH);  digitalWrite(pinG, HIGH);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    case 9:  // Set to 9  digitalWrite(pinA, HIGH);  digitalWrite(pinB, HIGH);  digitalWrite(pinC, HIGH);  digitalWrite(pinD, LOW);  digitalWrite(pinE, LOW);  digitalWrite(pinF, HIGH);  digitalWrite(pinG, HIGH);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  break;    }  }    void setup(){  Serial.begin(9600); // baud rate  // initialize digital pins as outputs  pinMode(pinA, OUTPUT);  pinMode(pinB, OUTPUT);  pinMode(pinC, OUTPUT);  pinMode(pinD, OUTPUT);  pinMode(pinE, OUTPUT);  pinMode(pinF, OUTPUT);  pinMode(pinG, OUTPUT);  // Digital pins  pinMode(D1, OUTPUT);  pinMode(D2, OUTPUT);  pinMode(D3, OUTPUT);  pinMode(D4, OUTPUT);  }  void loop(){  raw = analogRead(analogPin); // 0 - 5v mapped to 0 - 1023 bits at a rate of 4.9mV/unit    if(raw) { // Checks if analog pin has input  buffer = raw \* Vin;  Vout = (buffer)/1024.0; // 0 - 1023 bits  buffer = (Vin/Vout) -1;  //!< 500% percent error reduction  R2 = (R1 \* buffer) / 5;    if(R2 > 99999.0){ //!< Overload  // 3rd digit  digitalWrite(D3, HIGH);  lit(0);  delay(100); // internal synchronous clock waits for 0.001 second  // 4th digit  digitalWrite(D4, HIGH);  // Set to L  digitalWrite(pinA, LOW);  digitalWrite(pinB, LOW);  digitalWrite(pinC, LOW);  digitalWrite(pinD, HIGH);  digitalWrite(pinE, HIGH);  digitalWrite(pinF, HIGH);  digitalWrite(pinG, LOW);  // Prevents flickering  delay(100); // internal synchronous clock waits for 0.001 second  }      //!< Getting the digits  int num = (int)R2;  int r;  r = num % 10000;  digit1 = (num - r) / 10000;  num = r;  r = num % 1000;  digit2 = (num - r) / 1000;  num = r;  r = num % 100;  digit3 = (num - r) / 100;  num = r;  r = num % 10;  digit4 = (num - r) / 10;      //!< S7S display  for(int i = 0; i < 5; i++){  switch(i){  case 1:  digitalWrite(D1, HIGH);  lit(digit1);  case 2:  digitalWrite(D2, HIGH);  lit(digit2);  case 3:  digitalWrite(D3, HIGH);  lit(digit3);  case 4:  digitalWrite(D4, HIGH);  lit(digit4);  }  }    digitalWrite(D1, LOW);  digitalWrite(D2, LOW);  digitalWrite(D3, LOW);  digitalWrite(D4, LOW);  }  }  **Figure 3.** Arduino code programing the Serial 7-segment Display and Ohm-meter. |

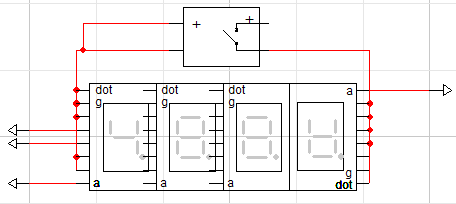
*Schematics:*



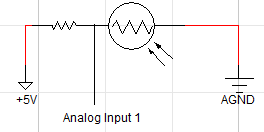
**Figure 4.** Bluetooth Optic Sensor and Buzzer circuit



**Figure 5.** Bluetooth Optic Sensor and Buzzer Simulation



**Figure 6.** Serial 7-Segment Display Schematic based off SH5461AS datasheet



**Figure 7.** Ohm meter schematic using a known resistor and photoresistor

*Voltage Table:*

**Table 1.** Voltage readings of the different components of the machine

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Photoresistor Voltage (V):** | **100k Resistor (V):** | **Buzzer (V):** | **LED (V):** | **1k Resistor (V):** | **MOSFET D-S (V):** | **MOSFET G-S (V):** |
| **Standby** | 0.00 | 0.07 | 0.05 | 0.16 | 0.00 | 0.14 | 0.00 |
| **Active** | 0.22 | 4.88 | 0.05 | Fluctuates | 0.00 | 2.82 | 0.23 |
| **LED on** | 2.70 | 0.77 | 0.05 | 2.60 | 2.41 | 0.00 | 3.39 |

*Photos:*

|  |  |
| --- | --- |
| **Figure 8.** Inside the machine    **Figure 9.** Seven-segment display wiring to the Arduino board    **Figure 10.** Arduino Genuino board that controls the seven-segment display | **Figure 11.** Arduino UNO board connected to a bluetooth chip    **Figure 12.** MOSFET, LED, Buzzer, and Photoresistor circuit    **Figure 13.** The group    **Figure 14.** The full machine |

Works Cited

"Arduino Bluetooth Basic Tutorial." *Arduino Project Hub*. N.p., n.d. Web. 08 May 2017. <<https://create.arduino.cc/projecthub/user206876468/arduino-bluetooth-basic-tutorial-d8b737>>.

"How to Make an Arduino Ohm Meter." *Circuit Basics*. N.p., 11 Apr. 2017. Web. 08 May 2017. <<http://www.circuitbasics.com/arduino-ohm-meter/>>.

Sapkota, Sagar. "Sagar Sapkota." *Build Circuit*. N.p., 12 Dec. 2015. Web. 08 May 2017. <<http://www.buildcircuit.com/darklight-sensor-using-transistor/>>.

Pinout. “SH5461AS Datasheet PDF – 4 Digit 7-Segment Display”. *Datasheet Cafe*. Web. 16 October 2015. <<http://www.datasheetcafe.com/sh5461as-datasheet-pdf/>>