University of Victoria
Department of Electrical Engineering & Computer Science
ECE 299 – Introduction to Electrical and Computer Engineering Design
Final Report
Summer Term 2020

Alarm Clock Project

August 18, 2020

Matthew Hermary
V00848259
Electrical Engineering
matthermary@gmail.com
August 18, 2020
B01

Branden Voss
V00913539
Electrical Engineering
brandenvoss97@hotmail.com
August 18, 2020
B01

Executive Summary

Electrical and Computer Engineering 299 is an introductory course to engineering design. In particular, this course is intended to familiarize students with the process of design in the professional world. Students have become oriented with skills such as creating electrical circuits, programming microcontrollers, planning and configuring printed circuit boards (PCB), basic knowledge of electronic components and industry practices [1]. As partial fulfillment of this course, teams were tasked with designing an alarm clock and implementing it into a PCB layout. The alarm clock must operate off of an Arduino Uno Rev 3 microcontroller and be capable of the following [2];

1. Display time in 24-hour format on 7-segment LED's
2. Allow time to be set or changed as desired
3. Allow alarm time to be set or disabled as desired
4. Trigger alarm of some form based on set time
5. Brightness of LED display to adjust according to ambient light
6. Optional snooze feature for alarm operation

Additionally, teams must present a 3-D model of an enclosure for the alarm clock developed through the use of software. The enclosure must be compatible to the alarm clocks dimensions and functionality [2]. One notable feature of the alarm clock presented in this report is the use of 2 7-segement decoders intended to free-up I/O pins on the Arduino. The subsequent report is of the topic of group 2, section B01's alarm clock and will develop its design, functionality and specifications in detail.

Table of Contents

List of Figures

List of Tables

Glossary

PCB	Printed Circuit Board.
LED	Light emitting diode. Electrical component that emits light

MOSFET	Metal oxide silicon field effect transistor. Used as a type of switch that allows current to flow from source to drain when a voltage is applied to its gate
Microcontroller	A small computer with memory and programmable input/output
KiCAD	Software featuring an integrated environment for schematic design and PCB layout

Project Goal

Through the use of techniques and topics discussed in ECE 299 and other applicable areas of study, teams of two are expected to design the layout of an alarm clock PCB and a compatible alarm clock enclosure. The end product must comply with requirements 1 through 5 outlined in the executive summary.

Constraints

This section of the report will discuss project specific constraints in addition to other unique constraints regarding the effects of COVID-19 and the ongoing global pandemic.

Project Specific Constraints

The following constraints are linked specifically to the project itself. Therefore, most of these constraints are hardware and software related. The first constraint to take note of is directly related to the hardware. In particular the use of an Arduino Uno Rev 3 microcontroller limited teams to the development of designs compatible with only this technology. The scope of this constraint in itself encompasses both software and hardware aspects. These include;

- Arduino output is limited to 40mA
- Number of pins on Arduino, both digital and analog available
- Constrained to Arduino programming language (C programming)

The Arduino limited teams to 13 digital pins and 6 analog pins. This may pose complications when considering all the functions incorporated into the alarm clock system. The Arduino time was to be advanced using a the millis function rather than delay statements which could not appropriately compensate time loss in a system

required to keep real time. Additionally, constraints were placed on the electronic components available for use under the TinkerCAD circuits software. It is important to also note that the dimensions of the enclosure and PCB must be suitable for an alarm clock design implementation. Branching from the latter constraint, the PCB layout would need to be reasonably compact.

Additional Constraints

Several more constraints that otherwise would not exist were identified due to the COVID-19 pandemic. Several complications were introduced regarding public health concerns and thus, ECE 299 in its entirety was required to be delivered online. Therefore, all instruction and information were given through online tools such as Zoom and Coursespaces to avoid face to face interaction. Also, all the hardware and programming was done through the TinkerCAD software as opposed to hands-on lab sessions. More specifically TinkerCAD restricted the availability of various electrical components and did not allow for the use of the Arduino clock library which may have provided alternative regarding software techniques.

Despite the given constraints throughout the design process both the course instructor and the lab instructors did an excellent job of recreating this course online in order to, as closely as possible, model the traditional learning experience.

Requirement Specifications

Upon considering the goals and given constraints, Group 2 from lab section B01 developed the following product in response to the project needs and requirements. The design process can be conveniently divided into separate parts for simplicity of explanation and understanding. Refer to

Appendix-B of this report for details on the alarm clock enclosure. The subsequent alarm clock design and functionality is expanded upon in this section of the report.

Project Planning

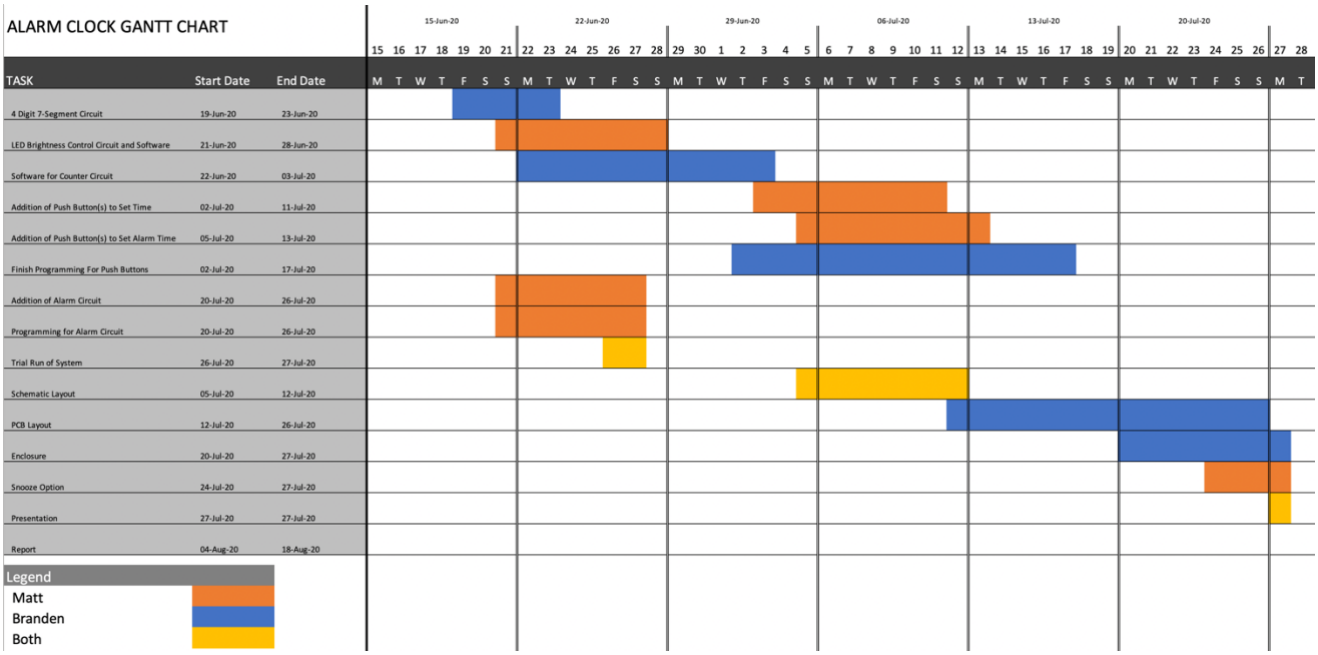
This section is intended to show how this project was managed through to completion by group 2 from lab section B01. A Gantt chart along with a bill of materials is presented.

Table 1. Bill of Materials [3]

Part Label	Manufacturer Part Number	Description	Unit Cost (CAD)	Quantity	Total Cost (CAD)
ARDUINO UNO	A000066	Arduino Uno Rev3 – 14 digital I/O pins, 6 analog inputs, USB connection, power jack and reset button.	$23.00	1	$23.00
7-segment_LED	SC56-11EWA	LED display and accessories HI EFF RED DIFFUSED 1 DIGIT	$1.90	4	$7.60
R_3	CF18JA50R	Carbon film general purpose resistor 50Ω	$0.16	1	$0.16
R_2	CF18JT1K50	Carbon film general purpose resistor 2.5kΩ	$0.16	1	$0.16
R_1	CF18JT75R	General purpose carbon film 75Ω resistor	$0.16	1	$0.16
Buzzer	7BB-12-9	9kHz Standard Buzzer Element 12mm diameter	$0.32	1	$0.32
U1	SN74LS47N	Designed for driving common cathode LEDs with active high outputs	$1.04	2	$2.08
SWx	RP3502MBRED	Pushbutton switch standard panel mount, two terminal	$5.39	5	$26.95
Q_Photo_NPN	TEPT5700	Phototransistor top view radial 570nm	$1.07	1	$1.07
NMOS_DGS	IRFZ44NPBF	MOSFET	$1.60	4	$6.40

The Gantt chart outlines task delegation from project start to finish. The legend in the bottom left corner colour codes tasks by group member.

Table 2. Gantt Chart



Circuit

The circuit designed by group 2 can be seen implemented using the KiCAD Schematics software in figure below. The circuit is operated using a single Arduino and utilizes four 7-segment LED displays branching from two separate 7-segment decoders. The circuit includes 5 pushbuttons to control alarm clock functions externally and a phototransistor configured to control the brightness of the LED display according to ambient light. Lastly, the system contains a single piezo buzzer to act as the output of the alarm function. This design makes use of both hardware components and software to operate properly. These technical characteristics are broken down into the following subsections.

Refer to Appendix-A for a complete listing of annotated code used for this project.

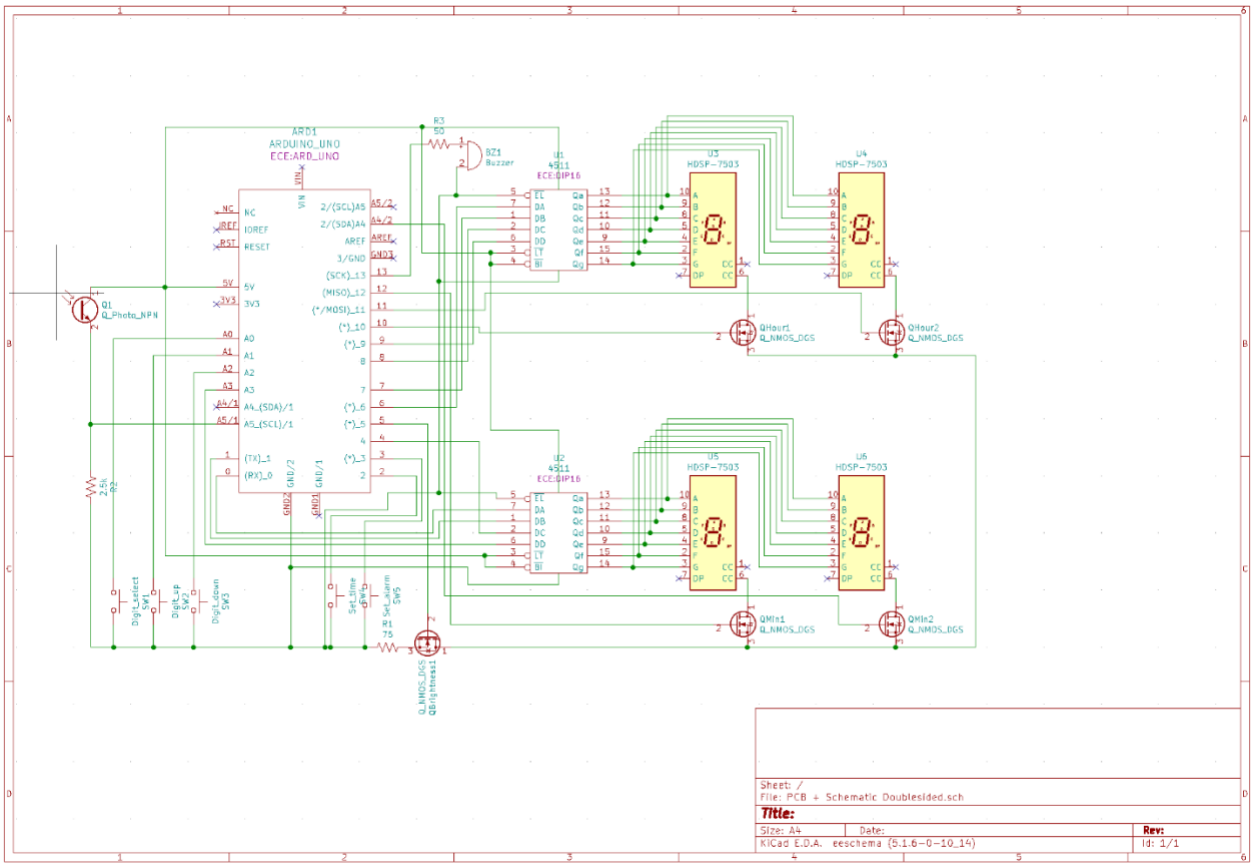


Figure 1. KiCAD schematic

7-Segment LED Display

The digits that hold time in hours (first two digits) on the LED display are operated from digital pins D6 through D9. The minute digits (last two digits) are operated from digital pins D0, D1, D4 and analog pin A3. These pins are configured as outputs that then go to their respective 7-segment decoders. Each set of four pins will produce a signal that represents a binary number to be displayed on the LED in base ten. In order to represent two separate numbers from the same set of 4 pins, a MOSFET is placed at the common cathode of each 7-segment display. In this manner, the MOSFET's are used a switches to select

which segment outputs the current number. Take for example, when the display is required to output ‘12’ on the hour digits the Arduino sends a binary ‘1’ to the decoder, then the base ten number ‘1’ is sent to both of the hour LED digits. From here the Arduino activates the MOSFET of the first digit to display the number. The same is done for the number ‘2’ which will be selected to display on the second digit. Every time the Arduino outputs a 1 or a 2 the corresponding MOSFET is switched on and the other off. This is done at a rapid rate so as to be unnoticed by the human eye. Since each segment is configured as common cathode, one current limiting resistor is placed at their common ground. This resistor is valued at 75Ω. The TinkerCAD implementation is shown below.

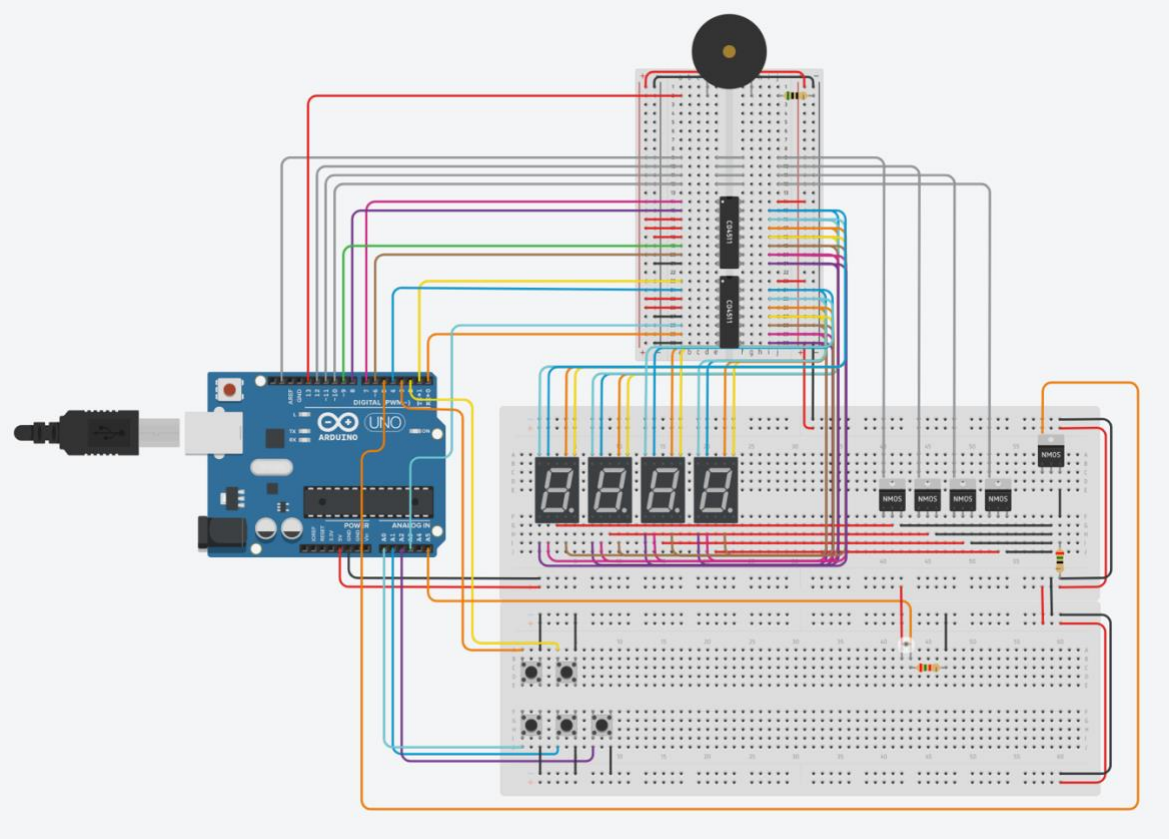


Figure 2. TinkerCAD implementation

Calculations

The designed circuit should guarantee that the digital I/O pins of the Arduino board carry between 10mA and 19mA of current. It is understood that the output voltage of the Arduino is approximately equal to 5V [4]. Additionally, the forward voltage of the LEDs are typically 1.9V but have a max of 2.3V [4]. The following design calculations are presented to aid in the understanding of the presented circuit. It is important to note a maximum of two 7-segment LEDs can display a number at any given time. I/O pin 2 supplies the first 7-segment LED. The following calculations justify the chosen resistor values:

$$R_{eq} = \frac{V - V_f}{I} \tag{1}$$

Where V is the voltage output of the Arduino, V_f is the forward voltage of the LED and I is the required current limit for each of the pins of the Arduino board 10mA.

$$R_{eq} = \frac{5\text{V} - 1.9\text{V}}{0.01A}$$

$$R_{eq} = 310\Omega$$

In order to display an "8", seven of the LED segments must but turned on. However, each LED shares the same resistor at their common ground. Therefore, the following calculations are used:

$$R_x = R_{eq} \cdot n \tag{2}$$

Where x in R_{1x} denotes a-f and n is the number or LEDs used in the segment.

$$R_{1x} = (310\Omega)(7)$$

$$R_{1x} = 2170\Omega$$

The resistance required in each of the inputs to the LED is 2170Ω. This holds true because of the following:

$$R_{eq} = (\frac{1}{R_{1a}} + \frac{1}{R_{1b}} + \frac{1}{R_{1c}} + \frac{1}{R_{1d}} + \frac{1}{R_{1e}} + \frac{1}{R_{1f}} + \frac{1}{R_{1g}})^{-1} \quad (3)$$

The above calculations are to be used for the remaining three LEDs. The calculated values for the remaining 7-segments is valid provided the values used in the calculations are relative to the LED being examined. Also, each resistor connected to the same 7-segment display is an equal portion of its R_{eq}.

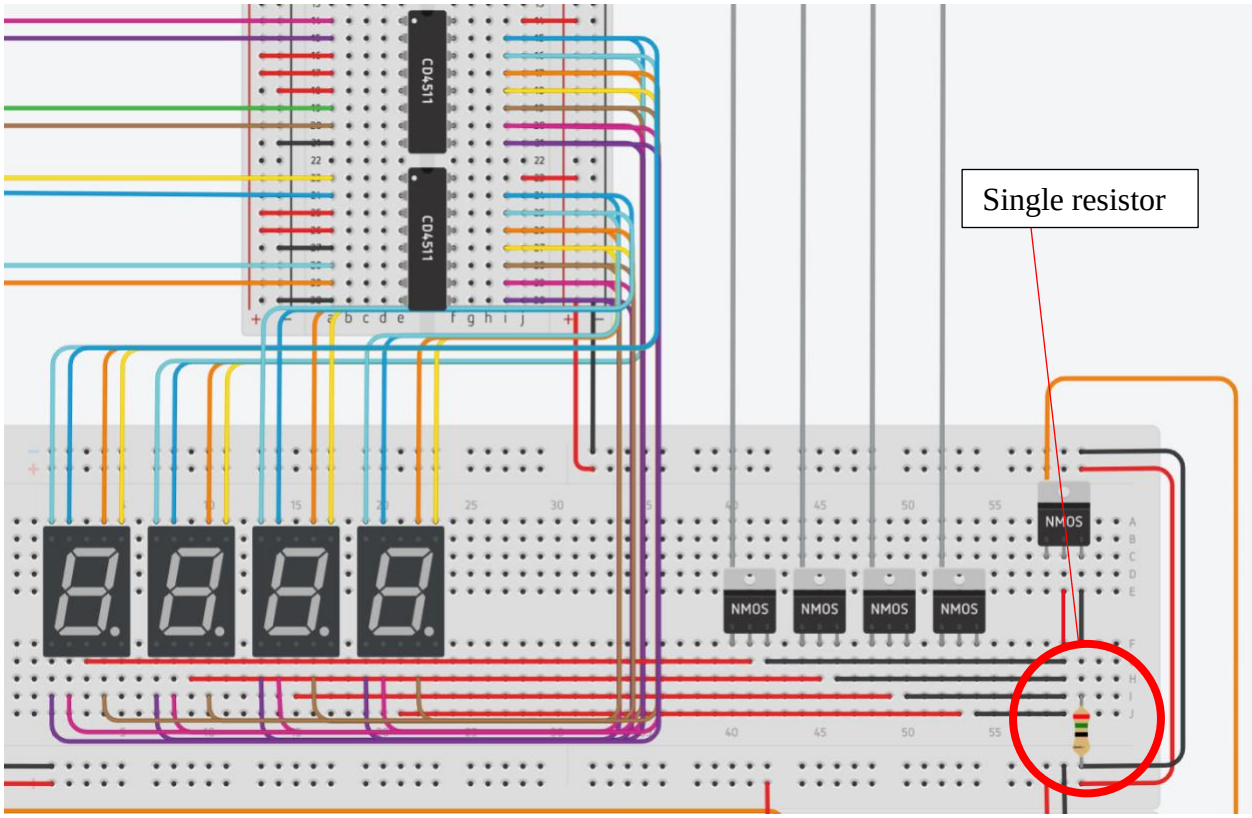


Figure 3. LED display circuit

Despite the calculated resistance value of 310Ω, the chosen value for the single current limiting resistor common to all four 7-segment displays is 75Ω. This was selected through trial and error and was observed to provide the optimal

brightness in our system. A resistance value of 310Ω was unsatisfactory and complications arose from the method used to display digits. The digits were not visibly clear and combinations of different digits were displayed on the segments.

Brightness

As specified in the project requirements, the LED display must orient its irradiance according to ambient light. The LED brightness is required to imitate the ambient light. Therefore, brighter ambient light corresponds to a brighter display; the inverse is also true. Current vs. irradiance is a linear relationship described by figure below. The current was measured in the 7-segment LEDS's using in ammeter and was found to have a range of 20mA under maximum ambient light conditions and 11mA for no light. 20mA corresponds to an irradiance of approximately 13mW/cm^2.

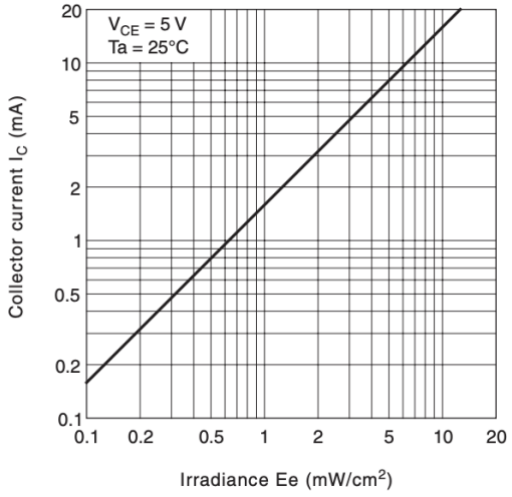


Figure 4. Chart of Irradiance vs Current

Push-Button Controls

The circuit consists of five pushbuttons; two of which are set up as interrupts and the remaining three are polled. The two interrupt buttons are responsible for telling the Arduino that the user would like to change the clock time and the alarm time respectively. Once one mode is selected the other three buttons will be polled by the Arduino. The first of the three buttons which is connected to A0 of the Arduino is necessary for selecting the digit to be changed. The following two

buttons will either increment or decrement the digit value. These pins are connected to A2 and A1 respectively.

Alarm Function

The alarm for this design is a single piezo buzzer. The buzzer is configured in a simple circuit to work with the programmed Arduino. The buzzer is connected to pin D13. The alarm will be triggered once the alarm set time matches the current time on the clock.

PCB

The previous circuit schematic developed using the KiCAD schematics software was then used to develop the printed circuit board layout. This PCB design is configured on a two layered board and is 116.586mm in length by 72.136mm in width. Figure shows the KiCAD PCB editor window. Some important measurements include [6];

- Track width of 10mils
- Via size of 40mils and drill of 20mils
- Ground track width of 20mils
- Ground via size of 45mils and drill of 25mils

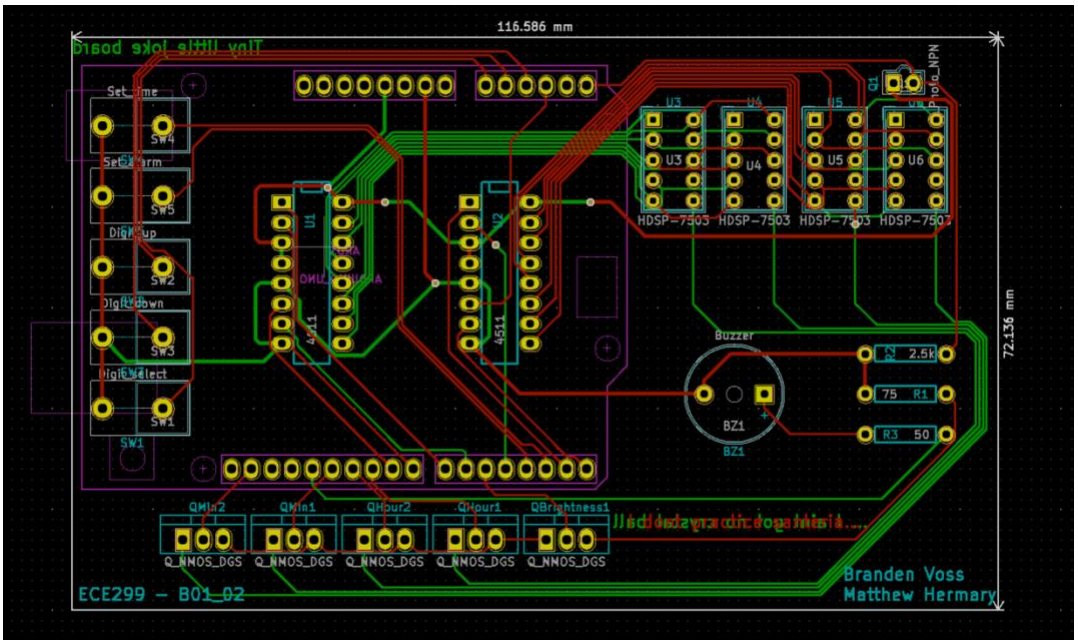


Figure 5. KiCAD PCB layout

These specifications are necessary to support the functions and capabilities of the Arduino Uno Rev 3 and its constituent circuit components.

Testing and Validation

This section of the report will discuss methods and procedures taken by group 2 in section B01 to test their design and validate their design elements and choices.

Display

The display testing was divided up into two parts: hardware and software where the wiring and capability of the decoder were tested, followed by testing of the function which called numbers to be displayed respectively.

Hardware Testing

To test the functionality of the 7-segment display decoder combination, inputs were hardcoded and fed into the decoder as their original binary numbers. During this time, no multiplexing of the displays was happening. For example, if 0001 and 0010 were fed into the hour and minute decoder respectively, the four displays would show 1122. Each bit was written with a digitalWrite high or low to the respective input pin.

Software Testing

Once the hardware was functioning properly, two identical functions were written: displayh and displaym for displaying hour and minutes respectively. These functions were tested similarly to the hardware, except the digitalWrite was handled within the function, and the only parameter passed through was a digit to be displayed. This was first tested by passing numbers directly, then were tested by passing the integer variables that contains the minute or hour digits for display.

Pushbutton interface

The two interrupt pushbuttons were designed to only illuminate the selected segment when in the function. For example, upon pushing the button segment 1 would be illuminated, and 2 to 4 would be dark. This meant that the alarm was either in the AlarmSet or TimeSet state depending on which button was pushed. To test the functionality of the other buttons, the selected digit was adjusted up or down to test them, afterwards, the last button was pressed which iterated to the next digit to be changed. If the SetTime function worked properly, the newly chosen time would be displayed. If the

SetAlarm function worked properly, the alarm would ring when the time matched the newly selected alarm time.

Ambient Light Sensing Subsystem

The ambient light sensing was done using voltage feedback from a simple phototransistor circuit. The voltage was read using an Analog pin and put out to the MOSFET using a Digital PWM pin. The Analog input reads in as values ranging from 0-1023, and the Digital PWM takes values from 0-255. This switch was done using the map function and the snippet of code *adc = map(adc, 45, 1010, 50, 255).* The input readings were displayed using Serial.println(adc) both before and after the conversion; this was printed while the simulation ran, and the phototransistor values were checked to see if they were modified and adjusted to the different ambient light. Once the values were shown to be adjusting and scaling properly, an ammeter was added to the circuit to ensure that the current was changing with different ambient light values.

Recommendations and Conclusion

Upon completion of this project, group 2 from section B01 successfully designed and implemented an alarm clock circuit and program. Extensive work was done to both test and develop the design under the project requirements. The implemented design successfully meets the goal specifications outlined in the executive summary. Furthermore, the alarm clock adequately holds 24-hour time along with an alarm function both of which can be adjusted to a desired value through external input. This alarm clock design also features a snooze option on the alarm. However, group 2 encountered some challenges when applying the phototransistor circuit to adjust display brightness according to ambient light.

Given a second chance to recreate this project in hindsight, there are some aspects that would be done differently. One particular aspect group 2 would address is to add resistors to each individual LED segment in order to better control the display brightness. This is due largely to the fact that different numbers use different segments and additional resistors will add some consistency to the current through the displays.

This project has given both team members the opportunity to develop a better understanding of the design process along with its limitations and challenges. It is with confidence that we can move forward with the benefits of learning from these challenges and adversities.

References

[1] 2019. Kingbright SC56-11EWA 14.22 Mm (0.56 Inch) Single Digit Numeric Display. 1st ed. Kingbright, pp.1-4.

[2] Mouser.com. 2020. Kingbright Leds Distributor | Mouser. [online] Available at: <https://www.mouser.com/manufacturer/kingbright/> [Accessed 30 May 2020].

[3] Store.arduino.cc. 2020. Arduino Uno Rev3 | Arduino Official Store. [online] Available at: <https://store.arduino.cc/usa/arduino-uno-rev3> [Accessed 30 May 2020].

[4] Digikey.ca. 2020. Digikey Electronics - Electronic Components Distributor. [online] Available at: <https://www.digikey.ca/?utm_adgroup=General&utm_source=google&utm_medium=cpc&utm_campaign=EN_Brand_General_E&utm_term=digikey%20canada&productid=&gclid=CjwKCAjw5cL2BRASEiwAENqAPt40mJnnZsI_PpX46xuYgrZwV4qnrwC9VARGciUG752xn3kX4dh06BoC6noQAvD_BwE> [Accessed 30 May 2020].

[5] I. Thirumarai. ECE 299. Class Lecture, Topic: “Project Planning” Electrical and Computer Engineering, University of Victoria, Victoria, BC, June 17, 2020.

[6] I. Thirumarai. ECE 299. Lab Report - Experiment 4, “ECE 299 – Intro to ECE Design” Lab Experiment 4 datasheet, June 25, 2020 [Accessed 25 June 2020].

[7] I. Thirumarai. ECE 299. Class Lecture, Topic: “Ambient light sensing using phototransistor” Electrical and Computer Engineering, University of Victoria, Victoria, BC, June 2, 2020.

[8] RS Components, “Light Dependent Resistors” 232 3616 datasheet, Mar. 1997 [Accessed 6 June 2020.

Appendix A – Annotated Code

The following is a complete listing of the code used in this project. The code is extensively commented and is made readily available in this section to whom it may interest.

```
const int m1 = 10;      //Sets pin 9 as a parameter for mosfet 1
const int m2 = 11;      //Sets pin 10 as a parameter for mosfet 2
const int m3 = 12;      //Sets pin 11 as a parameter for mosfet 3
const int m4 = 18;      //Sets pin 12 as a parameter for mosfet 4
const long interval = 10000;        //Interval of 1000 gives a second
unsigned long previousMillis = 0; //Millis variable to compare time elapsed
int min = 0;                        //2-digit minute value
int min_dig1;           //1-digit minute value holding the minute
int min_dig2;           //1-digit minute value holding the 10th minute
int hour = 0;                       //2-digit hour value
int hour_dig1 = 0;      //1-digit hour value holding hour
int hour_dig2 = 0;      //1-digit hour value holding the 10th hour
int time = 0;
int alarm_time = 0;
int alarm_min = 0;
int alarm_hour = 0;
bool alarm_on = true;
bool alarm_ringing = false;


void setup() //setup function maps pins
{
  pinMode (0, OUTPUT);              //Bit 0 on minute segments to decoder
  pinMode (1, OUTPUT);              //Bit 1 on minute segments to decoder
  pinMode (4, OUTPUT);              //Bit 2 on minute segments to decoder
  pinMode (A5, OUTPUT);             //Bit 3 on minute segments to decoder
  pinMode (6, OUTPUT);              //Bit 0 on hour segments to decoder
  pinMode (7, OUTPUT);              //Bit 1 on hour segments to decoder
  pinMode (8, OUTPUT);              //Bit 2 on hour segments to decoder
  pinMode (9, OUTPUT);              //Bit 3 on hour segments to decoder
  pinMode (10, OUTPUT);             //Output 10 controls NMOS 1
  pinMode (11, OUTPUT);             //Output 11 controls NMOS 2
  pinMode (12, OUTPUT);             //Output 12 controls NMOS 3
  pinMode (18, OUTPUT);             //Output 13 controls NMOS 4
  pinMode (13, OUTPUT);             //Output LED for testing alarm
  pinMode (A5, INPUT);              //Analog in for reading ambient light
  pinMode (5, OUTPUT);              //Analog write for PWM brightness on display
  pinMode (2, INPUT_PULLUP);    //Will trigger set time ISR
  pinMode (3, INPUT_PULLUP);    //Will trigger set alarm ISR
  pinMode (14, INPUT_PULLUP);  //Polling on A0 for input in time and alarm set
  pinMode (15, INPUT_PULLUP);  //Polling on A1 for input in time and alarm set
  pinMode (16, INPUT_PULLUP);  //Polling on A2 for input in time and alarm set
}

void select(int segm, int segh)
  //Takes in a number 1 or 2 and 3 or 4 and turns on that MOSFET for the displays
{
  switch(segm)
  {
    case 1:
    digitalWrite(m1, HIGH);
    digitalWrite(m2, LOW);
    break;

    case 2:
    digitalWrite(m1, LOW);
    digitalWrite(m2, HIGH);
    break;
```

```
   default:
   digitalWrite(m1, LOW);
   digitalWrite(m2, LOW);
   break;
  }
  switch(segh)
  {
   case 3:
   digitalWrite(m3, HIGH);
   digitalWrite(m4, LOW);
   break;

   case 4:
   digitalWrite(m3, LOW);
   digitalWrite(m4, HIGH);
   break;

   default:
   digitalWrite(m3, LOW);
   digitalWrite(m4, LOW);
   break;
  }
}

void displaym(int num)
//Displays a number in the minute segments
{
 switch(num)
 {
   case 0:
   //Illuminates segments to display "0"
   digitalWrite(0, LOW);          //input bit 0 low
   digitalWrite(1, LOW);          //input bit 1 low
   digitalWrite(4, LOW);          //input bit 2 low
   digitalWrite(A3, LOW);         //input bit 3 low
   break;


   case 1:
   //Illuminates segments to display "1"
   digitalWrite(0, HIGH);         //input bit 0 high
   digitalWrite(1, LOW);          //input bit 1 low
   digitalWrite(4, LOW);          //input bit 2 low
   digitalWrite(A3, LOW);         //input bit 3 low
   break;

   case 2:
   //Illuminates segments to display "2"
   digitalWrite(0, LOW);          //input bit 0 low
   digitalWrite(1, HIGH);         //input bit 1 high
   digitalWrite(4, LOW);          //input bit 2 low
   digitalWrite(A3, LOW);         //input bit 3 low
   break;

   case 3:
   //Illuminates segments to display "3"
   digitalWrite(0, HIGH);         //input bit 0 high
   digitalWrite(1, HIGH);         //input bit 1 high
   digitalWrite(4, LOW);          //input bit 2 low
   digitalWrite(A3, LOW);         //input bit 3 low
   break;

   case 4:
   //Illuminates segments to display "4"
   digitalWrite(0, LOW);          //input bit 0 low
```

```
   digitalWrite(1, LOW);          //input bit 1 low
   digitalWrite(4, HIGH);         //input bit 2 high
   digitalWrite(A3, LOW);         //input bit 3 low
   break;

   case 5:
   //Illuminates segments to display "5"
   digitalWrite(0, HIGH);         //input bit 0 high
   digitalWrite(1, LOW);          //input bit 1 low
   digitalWrite(4, HIGH);         //input bit 2 high
   digitalWrite(A3, LOW);         //input bit 3 low
   break;


   case 6:
   //Illuminates segments to display "6"
   digitalWrite(0, LOW);          //input bit 0 low
   digitalWrite(1, HIGH);         //input bit 1 high
   digitalWrite(4, HIGH);         //input bit 2 high
   digitalWrite(A3, LOW);         //input bit 3 low
   break;


   case 7:
   //Illuminates segments to display "7"
   digitalWrite(0, HIGH);         //input bit 0 high
   digitalWrite(1, HIGH);         //input bit 1 high
   digitalWrite(4, HIGH);         //input bit 2 high
   digitalWrite(A3, LOW);         //input bit 3 low
   break;


   case 8:
   //Illuminates segments to display "8"
   digitalWrite(0, LOW);          //input bit 0 low
   digitalWrite(1, LOW);          //input bit 1 low
   digitalWrite(4, LOW);          //input bit 2 low
   digitalWrite(A3, HIGH);        //input bit 3 high
   break;

   case 9:
   //Illuminates segments to display "9"
   digitalWrite(0, HIGH);         //input bit 0 high
   digitalWrite(1, LOW);          //input bit 1 low
   digitalWrite(4, LOW);          //input bit 0 low
   digitalWrite(A3, HIGH);        //input bit 3 high
   break;


   default: // no matches
   num = 0;
   break;
  }
}

void displayh(int num)
//Displays a number on the hour segments
{
  switch(num)
  {
   case 0:
   //Illuminates segments to display "0"
   digitalWrite(6, LOW);          //input bit 0 low
   digitalWrite(7, LOW);          //input bit 1 low
   digitalWrite(8, LOW);          //input bit 2 low
   digitalWrite(9, LOW);          //input bit 3 low
```

```
break;


case 1:
//Illuminates segments to display "1"
digitalWrite(6, HIGH);          //input bit 0 high
digitalWrite(7, LOW);           //input bit 1 low
digitalWrite(8, LOW);           //input bit 2 low
digitalWrite(9, LOW);           //input bit 3 low
break;

case 2:
//Illuminates segments to display "2"
digitalWrite(6, LOW);           //input bit 0 low
digitalWrite(7, HIGH);          //input bit 1 high
digitalWrite(8, LOW);           //input bit 2 low
digitalWrite(9, LOW);           //input bit 3 low
break;

case 3:
//Illuminates segments to display "3"
digitalWrite(6, HIGH);          //input bit 0 high
digitalWrite(7, HIGH);          //input bit 1 high
digitalWrite(8, LOW);           //input bit 2 low
digitalWrite(9, LOW);           //input bit 0 low
break;

case 4:
//Illuminates segments to display "4"
digitalWrite(6, LOW);           //input bit 0 low
digitalWrite(7, LOW);           //input bit 1 low
digitalWrite(8, HIGH);          //input bit 2 high
digitalWrite(9, LOW);           //input bit 3 low
break;

case 5:
//Illuminates segments to display "5"
digitalWrite(6, HIGH);          //input bit 0 high
digitalWrite(7, LOW);           //input bit 1 low
digitalWrite(8, HIGH);          //input bit 2 high
digitalWrite(9, LOW);           //input bit 3 low
break;


case 6:
//Illuminates segments to display "6"
digitalWrite(6, LOW);           //input bit 0 low
digitalWrite(7, HIGH);          //input bit 1 high
digitalWrite(8, HIGH);          //input bit 2 high
digitalWrite(9, LOW);           //input bit 3 low
break;


case 7:
//Illuminates segments to display "7"
digitalWrite(6, HIGH);          //input bit 0 high
digitalWrite(7, HIGH);          //input bit 1 high
digitalWrite(8, HIGH);          //input bit 2 high
digitalWrite(9, LOW);           //input bit 3 low
break;


case 8:
//Illuminates segments to display "8"
digitalWrite(6, LOW);           //input bit 0 low
digitalWrite(7, LOW);           //input bit 1 low
```

```
   digitalWrite(8, LOW);              //input bit 2 low
   digitalWrite(9, HIGH);             //input bit 3 high
   break;

   case 9:
   //Illuminates segments to display "9"
   digitalWrite(6, HIGH);             //input bit 0 high
   digitalWrite(7, LOW);              //input bit 1 low
   digitalWrite(8, LOW);              //input bit 2 low
   digitalWrite(9, HIGH);             //input bit 3 high
   break;

   default: // no matches
   num = 0;
   break;
  }
}

void set_alarm()
{
  int iter = 0;
  alarm_on == true;
  alarm_hour = time/100;
  alarm_min = time%100;
  //Set the alarm time
  while(iter == 0)
  {
    select(0, 4);            //Sets MOSFET 4 high others low
    //Adjust 10th hour
    if(digitalRead(15) == LOW)
    {
      //Increment hour by 10
      delay(100);
      alarm_hour = alarm_hour + 10;
      delay(100);
      if(alarm_hour >= 24)
      {
        alarm_hour = alarm_hour - 24;
      }
    }
    if(digitalRead(16) == LOW)
    {
      //Decrement hour by 10
      delay(100);
      alarm_hour = alarm_hour - 10;
      delay(100);
      if(alarm_hour < 0)
      {
        alarm_hour = alarm_hour + 24;
      }
    }

    //Display number
    displayh(alarm_hour/10);
    delay(100);

    if(digitalRead(14) == LOW)
    {
      delay(100);
      iter++;
      delay(100);
    }
  }//While(iter == 0)

  while(iter == 1)
  {
```

```
 select(0, 3);            //Sets MOSFET 3 high others low

 //Adjust hour
 if(digitalRead(15) == LOW)
 {
  //Increment hour by 1
  delay(100);
  alarm_hour++;
  delay(100);
  if(alarm_hour >= 24)
  {
   alarm_hour = alarm_hour - 24;
  }
 }
 if(digitalRead(16) == LOW)
 {
  //Decrement hour by 1
  delay(100);
  alarm_hour--;
  delay(100);
  if(alarm_hour < 0)
  {
   alarm_hour = alarm_hour + 24;
  }
 }

            //Display number
 displayh(alarm_hour%10);
 delay(10);

 if(digitalRead(14) == LOW)
 {
  delay(100);
  iter++;
  delay(100);
 }
}//While(iter == 1)

while(iter == 2)
{
 select(2, 0);            //Sets MOSFET 2 high others low
 //Adjust 10th minute

 if(digitalRead(15) == LOW)
 {
  //Increment minute by 10
  delay(100);
  alarm_min = alarm_min + 10;
  delay(100);
  if(alarm_min >= 60)
  {
   alarm_min = alarm_min - 60;
  }
 }
 if(digitalRead(16) == LOW)
 {
  //Decrement minute by 10
  delay(100);
  alarm_min = alarm_min - 10;
  delay(100);
  if(alarm_min < 0)
  {
   alarm_min = alarm_min + 60;
  }
 }
            //Display number
```

```
   displaym(alarm_min/10);
   delay(10);

   if(digitalRead(14) == LOW)
   {
     delay(100);
     iter++;
     delay(100);
   }
  }//While(iter == 2)

  while(iter == 3)
  {
   select(1, 0);            //Sets MOSFET 1 high others low

   //Adjust minute
   if(digitalRead(15) == LOW)
   {
     //Increment minute by 1
     delay(100);
     alarm_min++;
     delay(100);
     if(alarm_min >= 60)
     {
       alarm_min = alarm_min - 60;
     }
   }
   if(digitalRead(16) == LOW)
   {
     //Decrement minute by 1
     delay(100);
     alarm_min--;
     delay(100);
     if(alarm_min < 0)
     {
       alarm_min = alarm_min + 60;
     }
   }

              //Display number
   displaym(alarm_min%10);
   delay(10);
   if(digitalRead(14) == LOW)
   {
     delay(100);
     iter++;
     delay(100);
   }
  }//While(iter == 3)
  //Sets individual digits to new time
  alarm_time = alarm_hour*100 + alarm_min;
}

void set_time()
{
  int iter = 0;
  hour = time/100;
  min = time%100;
  //Set the time
  while(iter == 0)
  {
    select(0, 4);            //Sets MOSFET 4 high others low
    //Adjust 10th hour
    if(digitalRead(15) == LOW)
    {
      //Increment hour by 10
```

```
  delay(100);
  hour = hour + 10;
  delay(100);
  if(hour >= 24)
  {
    hour = hour - 24;
  }
 }
 if(digitalRead(16) == LOW)
 {
  //Decrement hour by 10
  delay(100);
  hour = hour - 10;
  delay(100);
  if(hour < 0)
  {
    hour = hour + 24;
  }
 }

 //Display number
 displayh(hour/10);
 delay(100);

 if(digitalRead(14) == LOW)
 {
  delay(100);
  iter++;
  delay(100);
 }
}//While(iter == 0)

while(iter == 1)
{
 select(0, 3);            //Sets MOSFET 3 high others low
 time = hour*100 + min;
 hour_dig1 = hour%10;
 hour_dig2 = hour/10;
 min_dig1 = min%10;
 min_dig2 = min/10;
            digitalWrite(13, LOW);//TESTING
 //Adjust hour
 if(digitalRead(15) == LOW)
 {
  //Increment hour by 1
  delay(100);
  hour++;
  delay(100);
  if(hour >= 24)
  {
    hour = hour - 24;
  }
 }
 if(digitalRead(16) == LOW)
 {
  //Decrement hour by 1
  delay(100);
  hour--;
  delay(100);
  if(hour < 0)
  {
    hour = hour + 24;
  }
 }

 //Display number
```

```
 displayh(hour%10);
 delay(10);

 if(digitalRead(14) == LOW)
 {
  delay(100);
  iter++;
  delay(100);
 }
}//While(iter == 1)

while(iter == 2)
{
 select(2, 0);            //Sets MOSFET 2 high others low
 //Adjust 10th minute

 if(digitalRead(15) == LOW)
 {
  //Increment minute by 10
  delay(100);
  min = min + 10;
  delay(100);
  if(min >= 60)
  {
   min = min - 60;
  }
 }
 if(digitalRead(16) == LOW)
 {
  //Decrement minute by 10
  delay(100);
  min = min - 10;
  delay(100);
  if(min < 0)
  {
   min = min + 60;
  }
 }

 //Display number
 displaym(min/10);
 delay(10);
 if(digitalRead(14) == LOW)
 {
  delay(100);
  iter++;
  delay(100);
 }
}//While(iter == 2)

while(iter == 3)
{
 select(1, 0);            //Sets MOSFET 1 high others low

 //Adjust minute
 if(digitalRead(15) == LOW)
 {
  //Increment minute by 1
  delay(100);
  min++;
  delay(100);
  if(min >= 60)
  {
   min = min - 60;
  }
 }
```

```
  if(digitalRead(16) == LOW)
  {
    //Decrement minute by 1
    delay(100);
    min--;
    delay(100);
    if(min < 0)
    {
      min = min + 60;
    }
  }

  //Display number
  displaym(min%10);
  delay(10);
  if(digitalRead(14) == LOW)
  {
    delay(100);
    iter++;
    delay(100);
  }
 }//While(iter == 3)
 //Sets individual digits to new time
 time = hour*100 + min;
 hour_dig1 = hour%10;
 hour_dig2 = hour/10;
 min_dig1 = min%10;
 min_dig2 = min/10;
}

void loop() //loop function
{
 unsigned long currentMillis = millis();
 if(currentMillis - previousMillis >= interval)
 {
  previousMillis = currentMillis;
  min++;

  if(min >= 60)
  {
    min = 0;              //Wraps minute around
    hour++;               //Increments hour
    if (hour >= 24)
    {
      hour = 0;           //Resets hour to zero on wraparound in 24hour mode
    }
    hour_dig1 = hour%10;
    hour_dig2 = hour/10;
  }
  min_dig1 = min%10;              //Calculates minute value
  min_dig2 = min/10; //Calculates 10th minute value
  time = hour*100 + min;
  //Check alarm step
  if(alarm_on == true &&  time == alarm_time)
  {
    buzzer();
  }

 }//End Timekeeper
 if(digitalRead(14) == LOW && alarm_on == true && alarm_ringing == true)
 {
  alarm_on = false;
  noTone(13);
  delay(100);
 }
```

```
 if(digitalRead(15) == LOW && alarm_on == true && alarm_ringing == true)
 {
  noTone(13);
  alarm_time = (alarm_time + 3);
  delay(100);
 }

 int adc = analogRead(5);
 adc = map(adc, 45, 1010, 50, 255);
 analogWrite(5, adc);
 attachInterrupt(digitalPinToInterrupt(2), set_time, FALLING);
 attachInterrupt(digitalPinToInterrupt(3), set_alarm, FALLING);
 select(1,3);
 displaym(min_dig1);
 displayh(hour_dig1);
 delay(10);

 select(2,4);
 displaym(min_dig2);
 displayh(hour_dig2);
 delay(10);
}

void buzzer()
{
 tone(13, 100);
 alarm_ringing = true;
}
```

Appendix B – Enclosure Details

Through the use of TinkerCAD 3-D, group 2 was able to develop a suitable enclosure for the alarm clock hardware. This model is displayed in figure. The enclosure is compatible to the alarm clock PCB and Arduino microcontroller both functionally and dimensionally. The model has a transparent front face intended to exhibit the PCB and circuitry inside the enclosure. Group 2 felt that it would be a unique design feature to display the work put into this project in this manner. The enclosure has through holes arranged in a rectangle allowing the PCB to be fixed. The spacer pegs allow the Arduino to sit between the PCB and the back wall of the model and an oval shaped hole is placed in the back in order for necessary power cables to be routed into the enclosure. Additionally, there are five vertical holes for the buttons that set both alarm and clock time as well as the alarm shut off and snooze option. The design also contains several small holes oriented in a radial fashion on the side wall intended to allow the alarm sound to be heard clearly outside the structure.

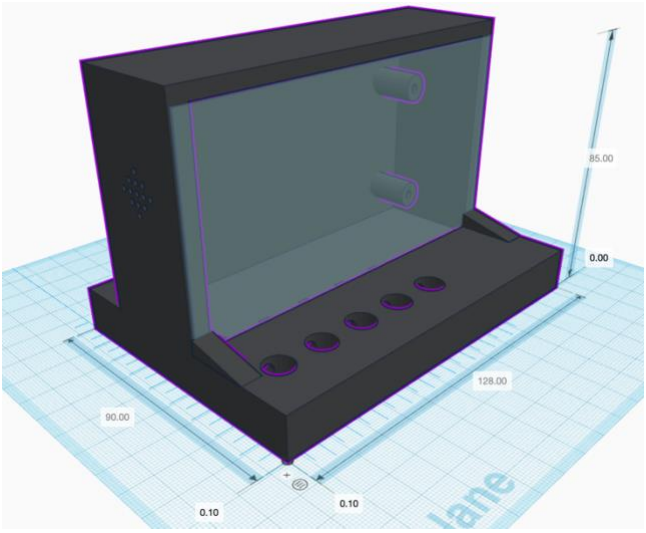


Figure 6. Alarm clock enclosure

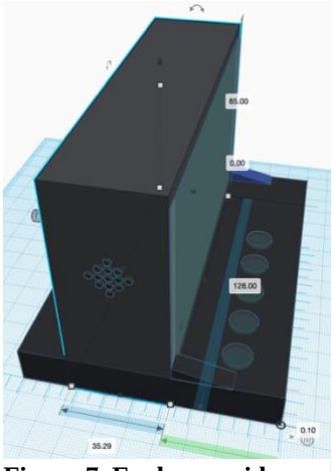


Figure 7. Enclosure side view

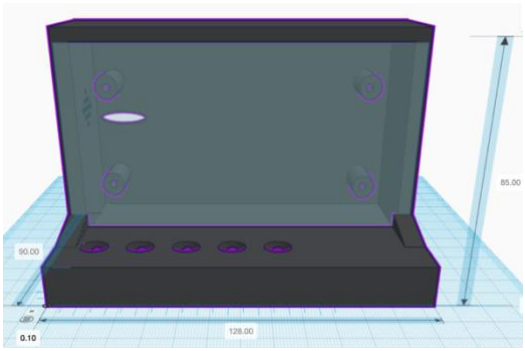


Figure 8. Enclosure front view