John Abbott College

Engineering Technologies Department

Name/Team Names:

Maya A. Rosales O.

Project 2

Technical Report

Title: Analog Clock with MP3 Player

Instructor: Hana Chammas

Date: May 27, 2023

Winter 2023

Contents

Introduction

For my project 2, I set out to build an analog clock that plays music. In this document, I will go into lengthy detail about the problem that I was trying to solve, the solution I came up with, and the steps I took to reach that solution.

I also describe in detail the process of building the clock, along with how to use it.

I wanted the clock to be aesthetically pleasing and intuitive to use and this is a philosophy I kept in mind while working on my project.

All the relevant files along with the bill of materials are also available in the index section of this document.

I would like to thank Nic the technician once again for all his help and support while working on this project.

Problem Description

With modern phones, most people have alarms at the palm of their hands, the issue with this is that phones can fail for a number or reasons. It can range from simply forgetting to charge one's battery, to it getting stolen, software malfunction, busted speak, etc. Having a separate dedicated time keeping/alarm device would allow for a better peace of mind and could make nice décor piece, enhancing the appearance of a room.

Most people are also woken up by blaring and disruptive alarm sounds. Although this does the job, it is rather unpleasant and, wouldn't a person prefer to wake up to the sound of their favorite song?

Proposed Solution

My proposal is to build an analog clock so that a user can upload their own tracks to play when the alarm goes off. The tracks will play in either a predefined sequence or at random, as selected by the user via an interface on the analog clock.

The reason it's analog as opposed to digital is that analog adds a certain aesthetic element to the project that a digital clock doesn't.

<u>Alternatives</u>:

Radio alarms have been around for a very long time, although this doesn't allow the user to choose the music being played.

There are also light alarms, which instead of playing a sound, will use a light source to attempt to wake up the person. This is an interesting alternative, as most of these alarms claim to have health benefits over regular sound alarms. The issue with this is that the science behind it remains contentious as best. Not only that, but most of these alarms can be rather expensive.

The pros and cons of using a phone have been discussed in the previous section.

Product Development

From the beginning, the idea was to create two separate systems for the clock, the “clock”, which would be handled by the PIC16F690 and the MP3 Player, which would be handled by an Arduino. The idea of creating two separate systems instead of one was so that they could operate independent of each other. This allows the user to listen to music using the MP3 player without disrupting the clock functionalities. Each microcontroller can “focus” on what it’s good at.

After this, I moved on to the electrical and mechanical aspect of the clock. For the electrical, I designed a schematic and PCB board using Diptrace. I then converted the PCB into Gerber files which I then sent to JLCPCB to print into a board.

The mechanical aspect involved 3D modeling the gear system and the case. After this was done, pieces were 3D printed and assembled. The front plate and the needles were laser cut and engraved.

Clock & Alarm

I’d like to preface by saying that a huge chunk of the time dedicated to this project was spent coding the PIC16F690. This is because no relevant libraries of code were readily available. Therefore, all the code for the clock and alarm was written from scratch.

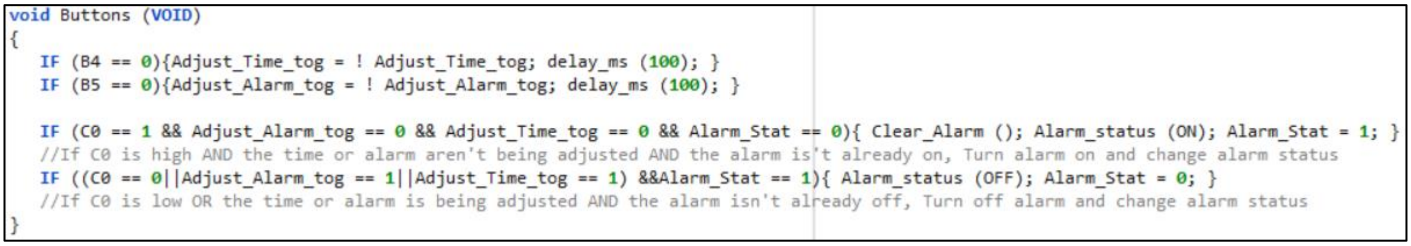
With the understanding that two stepper motors would be used to move the needles, a rotary encoder to control the motors, an RTC module to keep track of the time and a handful of switches and buttons as UI, these components were bought from the very beginning, allowing for prototyping to begin.

Very early in development, it was made very obvious that the microchip did not have enough pins for all the I/O’s that were required. This prompted the purchase of the MCP23008 I/O port expander.

The code works as follows:

On power up, the microcontroller will enable all the ports as inputs, enable I2C communication, turn the alarm off (if the alarm switch is on, the alarm will be turned on anyways) and move the hands to the 0 position.

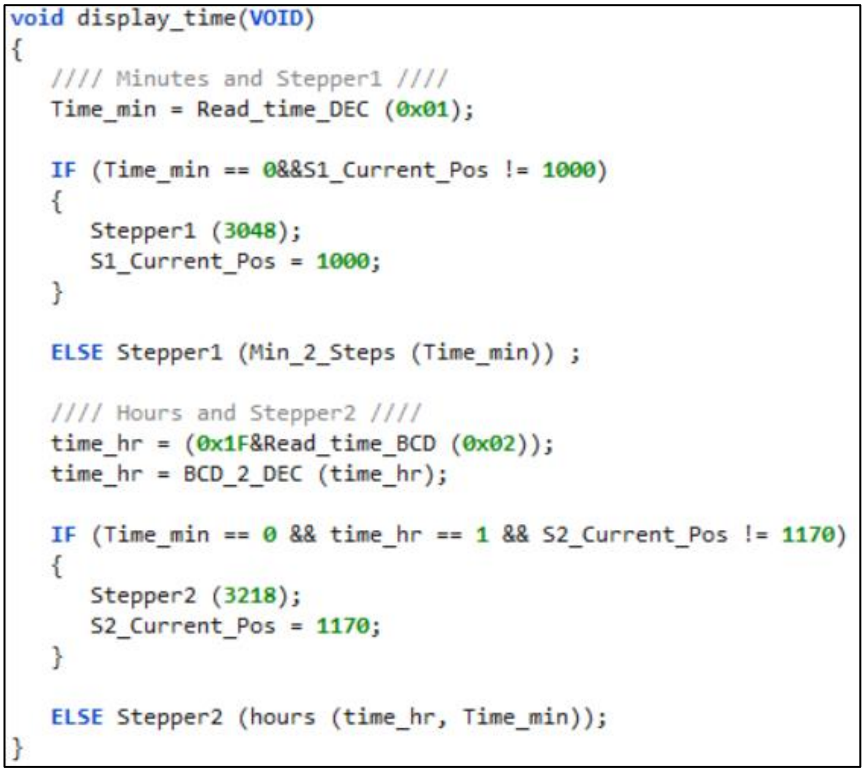
The first think the code does once it has been initialized is to check if any buttons have been pressed. The “Buttons()” function is called, which checks if the Time button, Alarm button or Alarm switch have been toggled.



```
void Buttons (VOID)
{
   IF (B4 == 0){Adjust_Time_tog = ! Adjust_Time_tog; delay_ms (100); }
   IF (B5 == 0){Adjust_Alarm_tog = ! Adjust_Alarm_tog; delay_ms (100); }

   IF (C0 == 1 && Adjust_Alarm_tog == 0 && Adjust_Time_tog == 0 && Alarm_Stat == 0){ Clear_Alarm (); Alarm_status (ON); Alarm_Stat = 1; }
   //If C0 is high AND the time or alarm aren't being adjusted AND the alarm is't already on, Turn alarm on and change alarm status
   IF ((C0 == 0||Adjust_Alarm_tog == 1||Adjust_Time_tog == 1) &&Alarm_Stat == 1){ Alarm_status (OFF); Alarm_Stat = 0; }
   //If C0 is low OR the time or alarm is being adjusted AND the alarm isn't already off, Turn off alarm and change alarm status
}
```

Once this is done, the code has three branches. Adjust time, Adjust Alarm and Display time. Since, in theory, the DS3231 already has the proper time stored, the clock will initialize on "Display time mode."

Display time

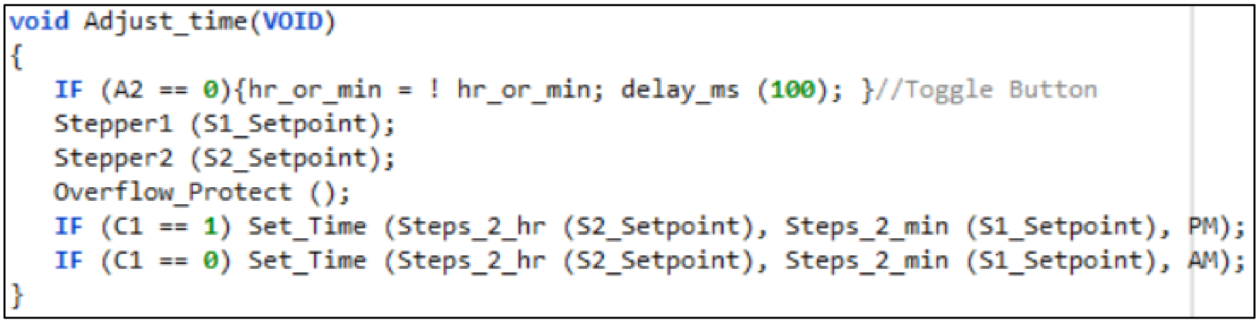


```
void display_time(VOID)
{
   //// Minutes and Stepper1 ////
   Time_min = Read_time_DEC (0x01);

   IF (Time_min == 0&&S1_Current_Pos != 1000)
   {
      Stepper1 (3048);
      S1_Current_Pos = 1000;
   }

   ELSE Stepper1 (Min_2_Steps (Time_min)) ;

   //// Hours and Stepper2 ////
   time_hr = (0x1F&Read_time_BCD (0x02));
   time_hr = BCD_2_DEC (time_hr);

   IF (Time_min == 0 && time_hr == 1 && S2_Current_Pos != 1170)
   {
      Stepper2 (3218);
      S2_Current_Pos = 1170;
   }

   ELSE Stepper2 (hours (time_hr, Time_min));
}
```

The display time function first fetches the time (minutes) from the DS3231 and stores the value in Time_min. Minutes are then converted to steps and the stepper is moved into positions. Although the stepper requires 2048 steps to make a full rotation, Minutes advance in 34 steps increments (60 mins = 2040 steps). This is because the microcontroller is unable to handle decimals and to get a an exact 2048, each min would have to be 34.13333 steps. To compensate for this the last step has an additional 8 steps.

The hours needle works in a similar manner. It fetches the time (hours) from the DS2321 and stores the value into Time_hr. It masks the first 3 bits as they contain information that is not relevant and then converts it into decimal. Hours increase in 170 step increments. The hours needle also moves slightly closer to the next hour as minutes increase. This is done to simulate an analog clock. For every 6 mins, the clock will move 17 steps. Once again, the hour of the clock has an additional 8 steps to compensate.

Adjust time



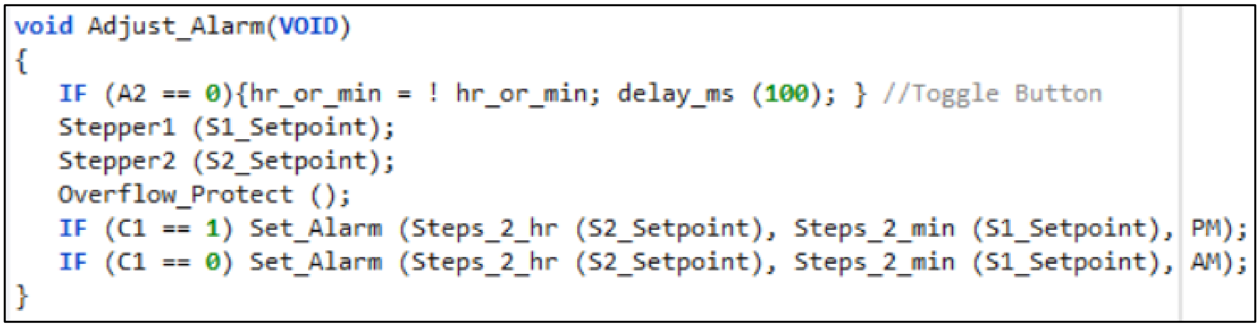
```
void Adjust_time(VOID)
{
   IF (A2 == 0){hr_or_min = ! hr_or_min; delay_ms (100); }//Toggle Button
   Stepper1 (S1_Setpoint);
   Stepper2 (S2_Setpoint);
   Overflow_Protect ();
   IF (C1 == 1) Set_Time (Steps_2_hr (S2_Setpoint), Steps_2_min (S1_Setpoint), PM);
   IF (C1 == 0) Set_Time (Steps_2_hr (S2_Setpoint), Steps_2_min (S1_Setpoint), AM);
}
```

The adjust time function first enables interrupts to allow control of the motors through the rotary encoder. After this, it will check if the built in button on the rotary encoder has been pressed. This button allows the user to toggle between motors. After this, the two steppers will be moved into the new position.

The “Overflow_Protect” function serves to keep the position value of the motors within the expected range. The code is designed under the assumption that motor is within 0-2048 steps. If the motor is outside this range, it will replace the position value with an equivalent position value. If the motor goes to step 2148, It will move to position 2148 and replace the position with 100 (2148 - 2048 = 100).

Finally, the new position will be converted into time and the DS3231 will be updated. Whether the time is AM or PM is determined by the position of the AM/PM switch.

Adjust Alarm

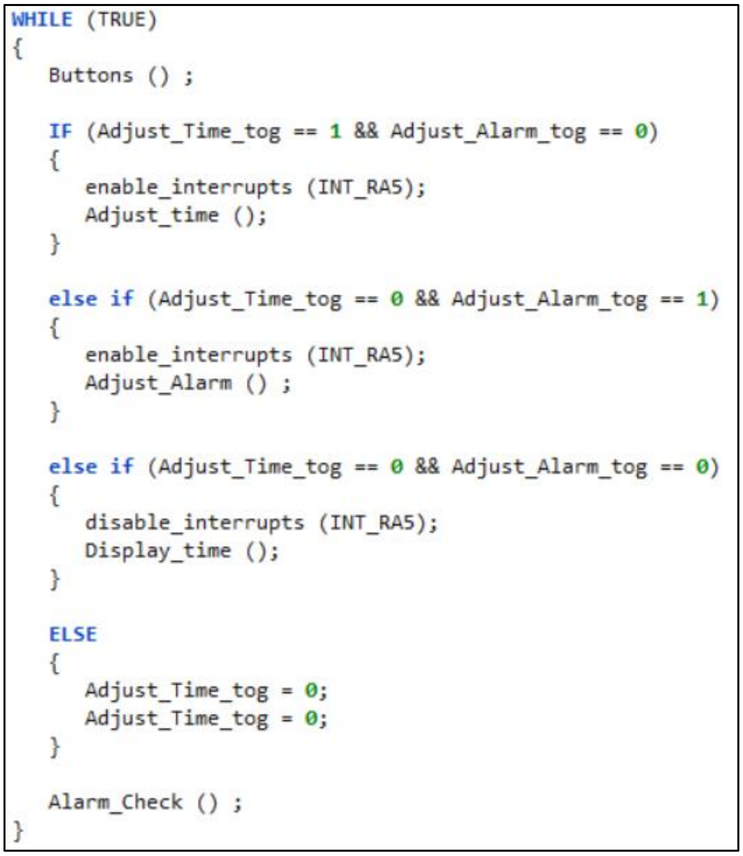


```
void Adjust_Alarm(VOID)
{
   IF (A2 == 0){hr_or_min = ! hr_or_min; delay_ms (100); } //Toggle Button
   Stepper1 (S1_Setpoint);
   Stepper2 (S2_Setpoint);
   Overflow_Protect ();
   IF (C1 == 1) Set_Alarm (Steps_2_hr (S2_Setpoint), Steps_2_min (S1_Setpoint), PM);
   IF (C1 == 0) Set_Alarm (Steps_2_hr (S2_Setpoint), Steps_2_min (S1_Setpoint), AM);
}
```

The adjust alarm function operates the same the adjust time function does, but instead of updating the time register in the DS3231, it updates its alarm register.

Lastly, the code will check if the alarm has gone off and if so, it will clear the alarm to allow for the next time I needs to go off.

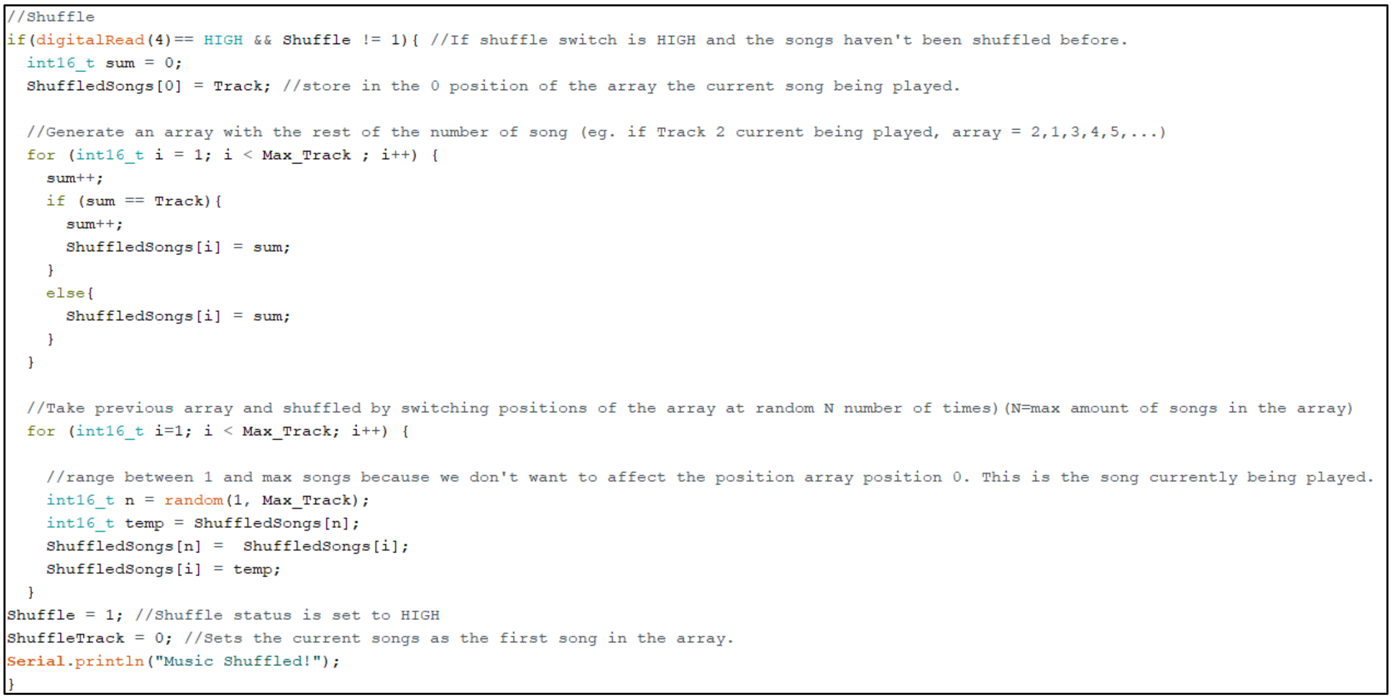
The main function looks as shown bellow, the full code along with the relevant header files are available in the manufacturing section of this document.



```
WHILE (TRUE)
{
   Buttons () ;

   IF (Adjust_Time_tog == 1 && Adjust_Alarm_tog == 0)
   {
      enable_interrupts (INT_RA5);
      Adjust_time ();
   }

   else if (Adjust_Time_tog == 0 && Adjust_Alarm_tog == 1)
   {
      enable_interrupts (INT_RA5);
      Adjust_Alarm () ;
   }

   else if (Adjust_Time_tog == 0 && Adjust_Alarm_tog == 0)
   {
      disable_interrupts (INT_RA5);
      Display_time ();
   }

   ELSE
   {
      Adjust_Time_tog = 0;
      Adjust_Time_tog = 0;
   }

   Alarm_Check () ;
}
```

MP3 Player

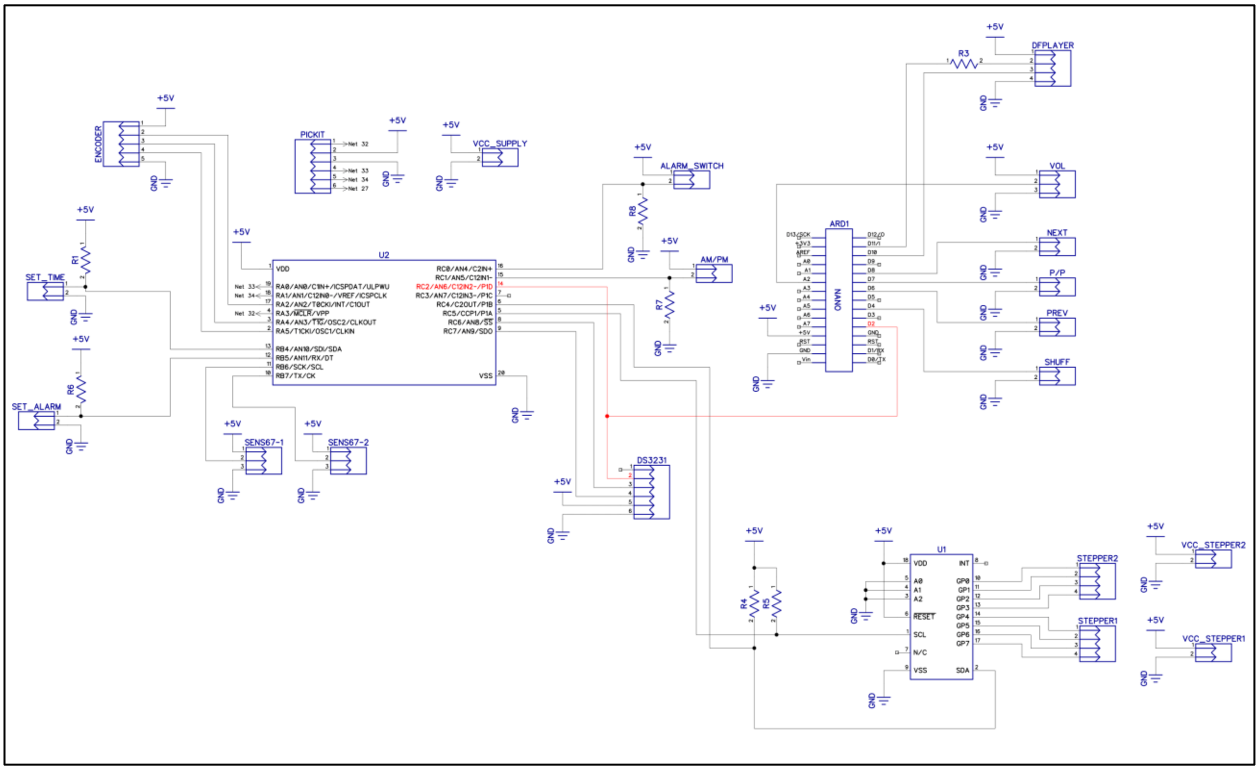
The Arduino code was done considerably faster than the PIC16F690's code. This is because the DFPlayer (The MP3 module that handles the musical aspect of this project), comes with pre-made functions for the Arduino, available at the manufacturers' website. Pressing the various control buttons simply calls these pre-made functions. My main contribution was creating a shuffle function. It operates by generating an array of all the songs and then proceeding to shuffle that array back and forth.

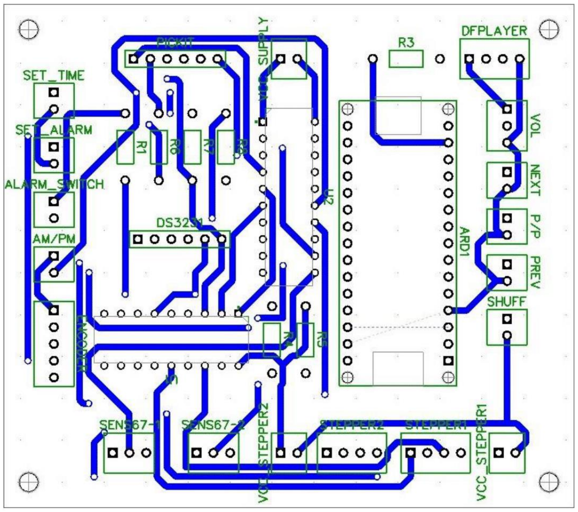


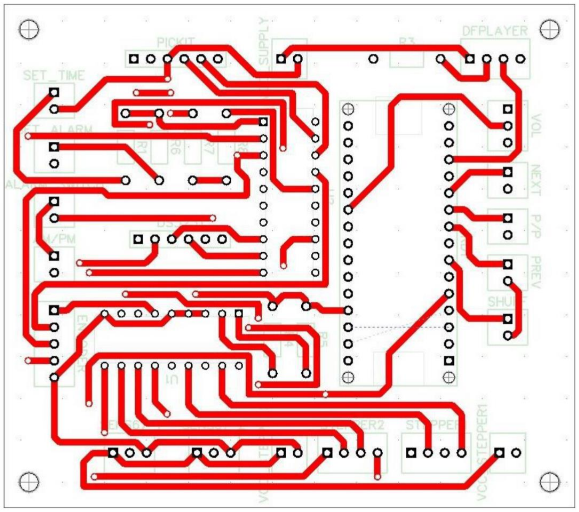
```
//Shuffle
if(digitalRead(4)== HIGH && Shuffle != 1){ //If shuffle switch is HIGH and the songs haven't been shuffled before.
  int16_t sum = 0;
  ShuffledSongs[0] = Track; //store in the 0 position of the array the current song being played.

  //Generate an array with the rest of the number of song (eg. if Track 2 current being played, array = 2,1,3,4,5,...)
  for (int16_t i = 1; i < Max_Track ; i++) {
    sum++;
    if (sum == Track){
      sum++;
      ShuffledSongs[i] = sum;
    }
    else{
      ShuffledSongs[i] = sum;
    }
  }

  //Take previous array and shuffled by switching positions of the array at random N number of times)(N=max amount of songs in the array)
  for (int16_t i=1; i < Max_Track; i++) {

    //range between 1 and max songs because we don't want to affect the position array position 0. This is the song currently being played.
    int16_t n = random(1, Max_Track);
    int16_t temp = ShuffledSongs[n];
    ShuffledSongs[n] =  ShuffledSongs[i];
    ShuffledSongs[i] = temp;
  }
Shuffle = 1; //Shuffle status is set to HIGH
ShuffleTrack = 0; //Sets the current songs as the first song in the array.
Serial.println("Music Shuffled!");
}
```

Schematics and Board

After fully coding the project and having a working prototype, I moved on to creating a compact PCB board for all the components. Although the electrical aspect of this project is not exceedingly complex, creating a schematic and a board is still crucial as there are several connections on the board and multiple inputs and outputs. Attempting to create a simple board from a prototype PCB would be excruciatingly hard. This is why once I had a schematic and a PCB board design, I sent it out to print with JLCPCB. The schematic and the PCB are as shown bellow and the files are available in the manufacturing section of this document.





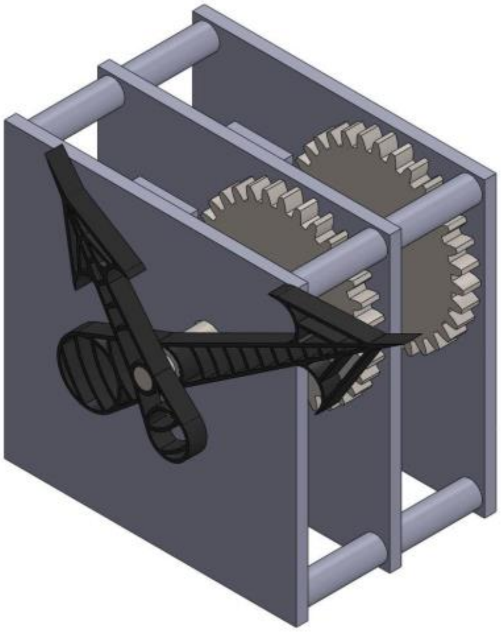


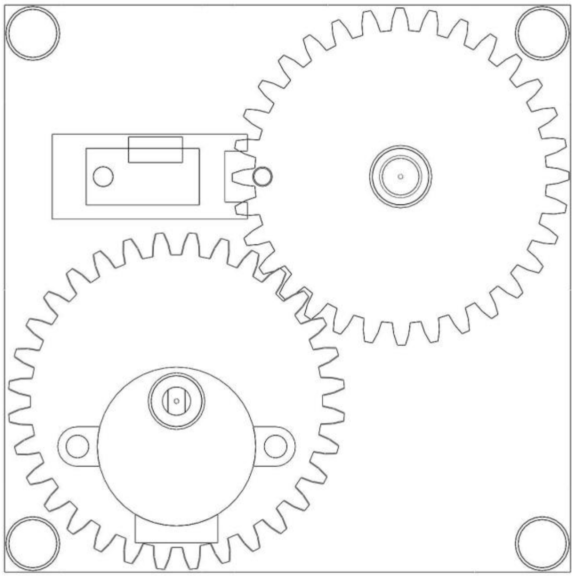
Gearbox

After handling the software and electrical aspects of the project. The mechanical aspect came into focus. Clocks generally have concentric shafts. Each one controlling a single needle. The goal was to recreate this system using the motors that I had in hand. The “gearbox” as shown bellow was conceived.

Each gear has 32 teeth. 32 being a divisor of 2048 which is the total amount of steps for a full revolution.

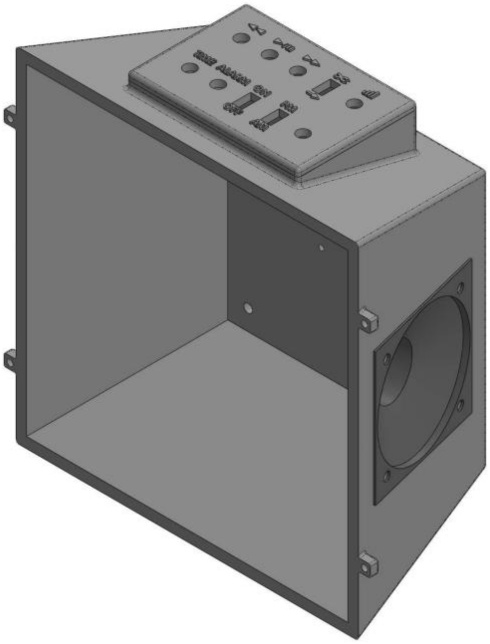
On both of the gears that function as shafts, there is a small whole near the edge of the gear. Inside this whole there’s a magnet glued. This is because there is a hall sensor module that hovers slightly above the gear. This sensor is used to determine the house position of the needles. On power up, the gears will rotate CCW until the magnet is detected and the home position is reached.

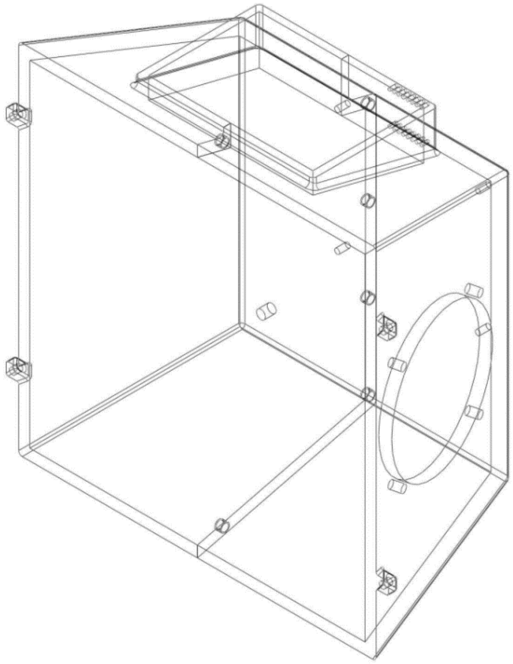




Case

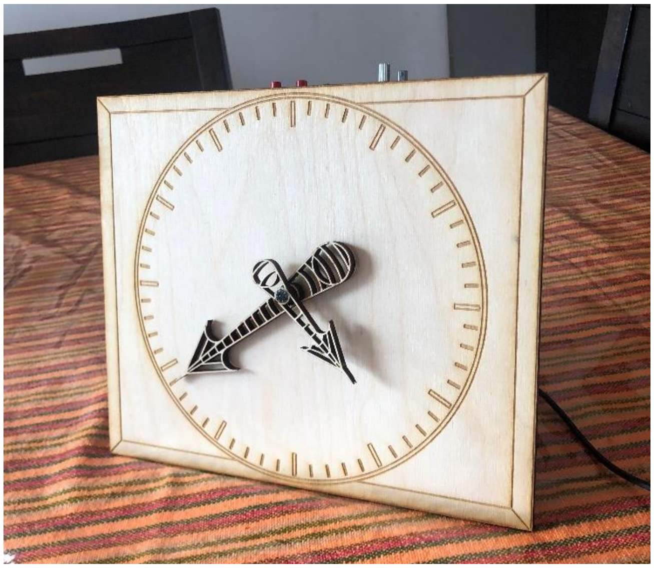
After this, the design of the case was rather simple. It simply had to be designed to fit the components and the gear box at the same time. Having a 3D model made of solid works with all the necessary references took out all the guesswork from this. The final look of the case is as shown bellow. Although because of its size, it had to be printed in two parts and then glued together. The face plate and the needles were then laser cut and engraved at the Pierrefonds library.





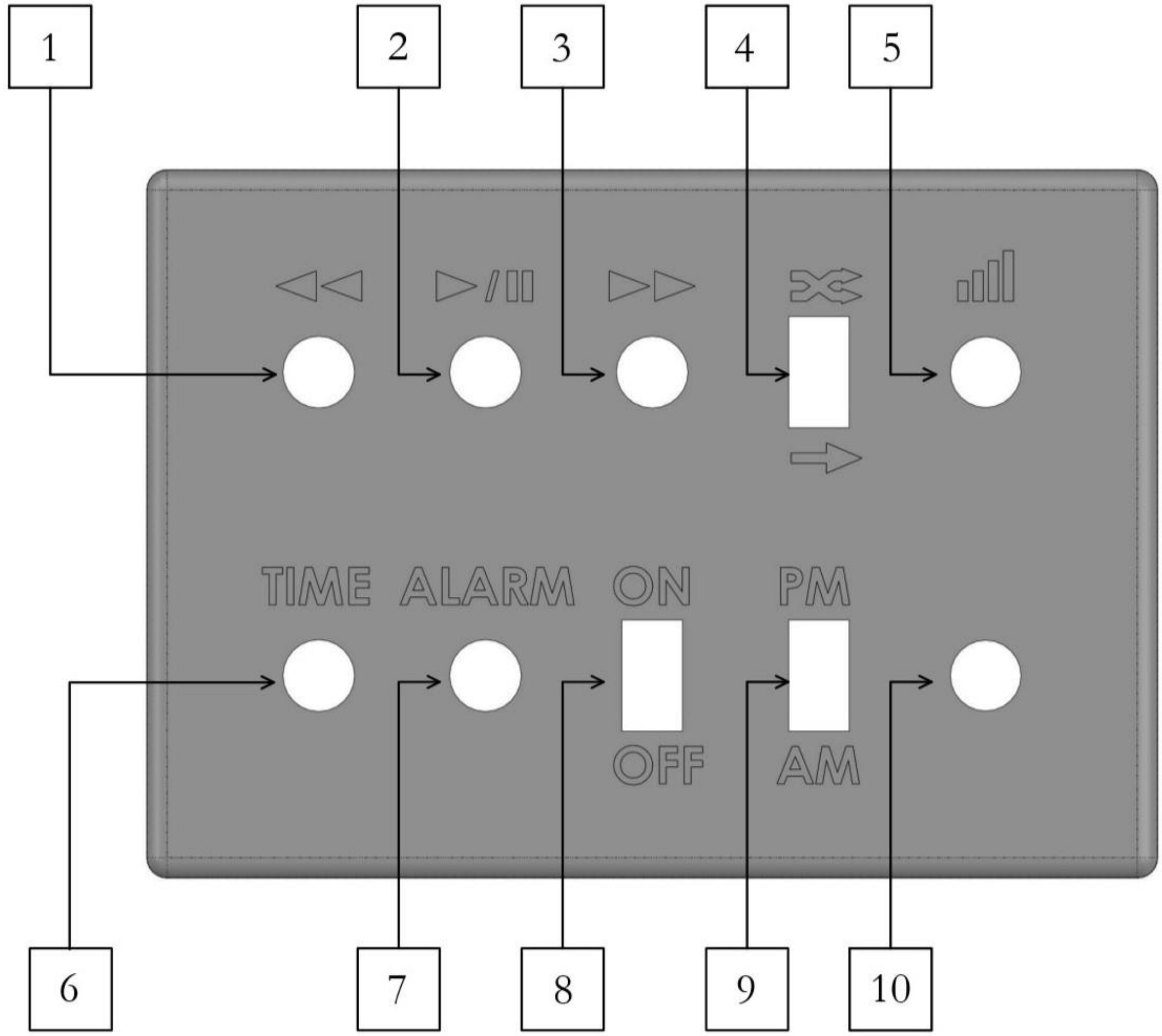
Final product.

A demo of the functional product is available in the media part of this document under “proof of work.” Various photos of the prototyping and building process are also available in the media section of this document.



Technical Product Description

Operation



1	**Previous Button** Pushing this button plays the previous song on the queue.	6	**Set Time Button** Pushing this button allows you to set the time using the AM/PM switch and the control knob.
2	**Pause Button** Pushing this button will pause the song currently playing. Pressing it again resumes it.	7	**Set Alarm Button** Pushing this button allows you to set the alarm using the AM/PM switch and the control knob.

3	**Next Button** Pushing this button plays the next song on the queue	8	**Alarm Switch** The alarm switch is used to turn the alarm on and off.
4	**Shuffle Switch** When the switch is on the shuffle position, all the songs inside of the SD card will be shuffled. If not, they will play in the order in which they are stored. When the alarm goes off, if the switch is in the shuffle position, it will play a song at random. If not, it will play the first song stored in the SD card.	9	**AM/PM Switch** As the clock only has 12h and the day has 24, the AM and PM switch helps the clock differentiate between before noon and after noon, whenever the user is setting the time or the alarm.
5	**Volume Control Knob** Turning the knob CW will increase the volume. Turning it CCW will decrease it.	10	**Clock Control Knob** The control knob is used to move the needles and set the time when changing the time or the alarm. The knob also functions as a button. Pushing it down will toggle control between the two needles.

MP3 Module

The MP3 module at the back of the control panel is for micro-SD cards. When plugging in the clock for the first time, an SD card ought to be inserted for the MP3 module to properly initialize.

Inside of the SD card, there needs to be a folder labeled as “MP3”, this is where songs should be stored for the module to recognize them.

The file names need to be formatted as follows:

0001 songtitle1
0002 songtitle2
0003 songtitle3
…

The module is able to handle up to 500 songs.

Maintenance

Inside of the clock, there is a DS3231 RTC module equipped with a LIR2032. This battery is what allows the clock to keep track of the time even when it isn’t powered. This battery should be replaced once every 10 years. It is important to know that one should **NOT** replace this battery with the more widely available CR2032. Doing so will make the battery expand.

An expanded battery is a fire and explosive hazard.

The clock should also not be stored in humid environments or direct sunlight. Direct sunlight may cause the PLA to shrink, warm and eventually crack. Humid environments may damage the wood face plate and the needles.

Troubleshooting

The internal machinery of the clock can be easily accessed by removing the four screws at the back of the face plate. Here are some simple troubleshooting steps for common issues. If any other issues arise, please contact the manufacturer for further assistance.

Rattling Noise at Loud Volumes

If there seems to be rattling noises at high volumes, it may be because the wires to the control panel are in contact with the back of the speaker. This causes the wires to rattle and make a noise at high volumes. The solution is simply to open the clock and shift the cables away from the speaker.

Clock Out of Phase

If the clock seems to be displaying the wrong time or is out of phase, simply unplug and plug back in the clock. The home position is reset on power up every time, so this ought to fix the issue, if not the home position might be out of phase.

Home Position out of Phase.

If the home position is out of phase, it is because the hall sensors that are used to set the home position have been tampered with. To fix this, simply access the inside of the clock. The gearbox will be attached to the back of the faceplate. Inside of the gearbox, there will be two red modules directly above the gears. To fix this issue, simply push the sensors back into position. Pushing the sensors CW will shift the home position CCW. Pushing the sensors CCW will shift the home position CW.

Clock Does Not Update Time On Power up

The clock is equipped with a module that allows it to keep track of the time even when it is disconnected. This allows the clock to auto set the time when powered on. If this is not the case, the battery might have died, consult the maintenance section of this document for further details.

MP3 Not Working

The MP3 will enter a “locked” state if the clock is initially powered without an SD card inserted. Try to unplug, insert an SD card, and plug in the device again. You should see the blue LED of the module flash once. For any other issues with the MP3 module, consult the operation section in this document.

Lessons Learned

I have learned about the advantages and disadvantages of different manufacturing processes.

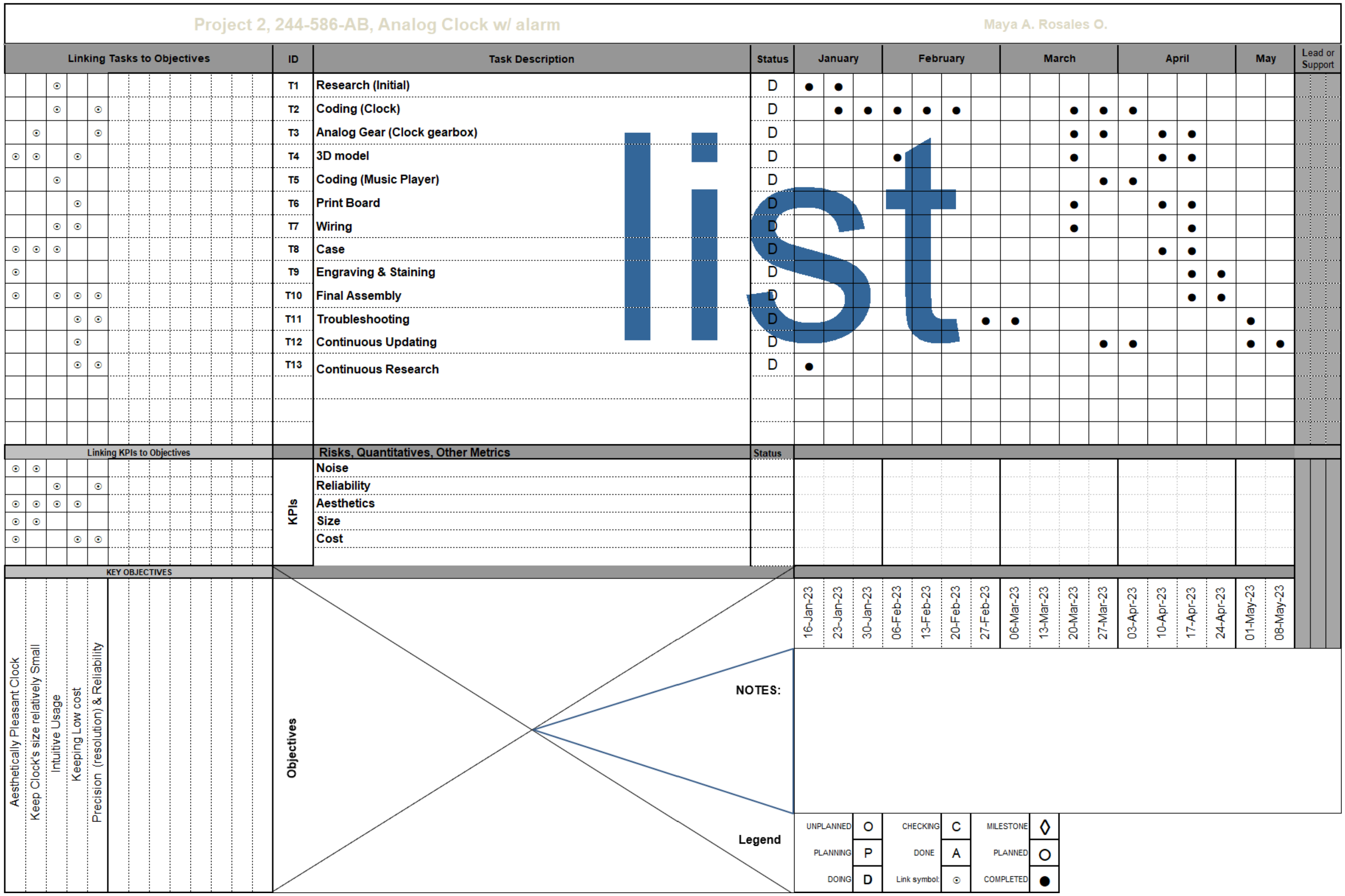
For the most part, I have stuck with 3D printing when it comes to building my projects. This is because 3D printing is accessible, relatively fast, and easy. In this project, I had the chance to work with a CNC engraving machine and it opened my eyes to new possibilities. CNC machines are simply much more precise than 3D printers. They also don't face the same limitations of wrapping and shrinkage. If given the chance to do it again, I would probably redo the gears with a CNC machine. The printed gears currently implemented have barely passable clearances. CNC would allow the gears to make more precise motions, making the clock more precise.

I also learned the hard way that PLA is a lot more susceptible to warping than I thought. I spray painted a piece and left it outside to dry. When I came back, the piece was warped.

3D printing still has its place though. I would not have been able to manufacture some of the more complex pieces through CNC. The shape of the case required the piece to be 3D printed.

Regarding project planning, I've learned that all project development loosely follows the same steps. Design. Prototype. Test. Rinse and repeat, until the project is complete.

Final OPPM



Project 2, 244-586-AB, Analog Clock w/ alarm

Maya A. Rosales O.

Linking Tasks to Objectives					ID	Task Description	Status	January			February				March				April				May		Lead or Support
Aesthetically Pleasant Clock	Keep Clock's size relatively Small	Intuitive Usage	Keeping Low cost	Precision (resolution) & Reliability				16-Jan-23	23-Jan-23	30-Jan-23	06-Feb-23	13-Feb-23	20-Feb-23	27-Feb-23	06-Mar-23	13-Mar-23	20-Mar-23	27-Mar-23	03-Apr-23	10-Apr-23	17-Apr-23	24-Apr-23	01-May-23	08-May-23	
		⊙			T1	Research (Initial)	D	●	●																
		⊙		⊙	T2	Coding (Clock)	D		●	●	●	●	●				●	●	●						
	⊙			⊙	T3	Analog Gear (Clock gearbox)	D										●	●		●	●				
⊙	⊙		⊙		T4	3D model	D				●						●			●	●				
		⊙			T5	Coding (Music Player)	D											●	●						
			⊙		T6	Print Board	D										●			●	●				
		⊙	⊙		T7	Wiring	D										●				●				
⊙	⊙	⊙			T8	Case	D													●	●				
⊙					T9	Engraving & Staining	D														●	●			
⊙		⊙	⊙	⊙	T10	Final Assembly	D														●	●			
			⊙	⊙	T11	Troubleshooting	D							●	●								●		
			⊙		T12	Continuous Updating	D											●	●				●	●	
			⊙	⊙	T13	Continuous Research	D	●																	

Linking KPIs to Objectives						Risks, Quantitatives, Other Metrics	Status
⊙	⊙				KPIs	Noise	
		⊙		⊙		Reliability	
⊙	⊙	⊙	⊙			Aesthetics	
⊙	⊙					Size	
⊙			⊙	⊙		Cost	

KEY OBJECTIVES

- Aesthetically Pleasant Clock
- Keep Clock's size relatively Small
- Intuitive Usage
- Keeping Low cost
- Precision (resolution) & Reliability

Objectives

NOTES:

Legend

UNPLANNED	O	CHECKING	C	MILESTONE	◊
PLANNING	P	DONE	A	PLANNED	○
DOING	D	Link symbol:	⊙	COMPLETED	●

References

[1]
“PIC16F690,” *Microchip.com*, 2023. https://www.microchip.com/en-us/product/PIC16F690 (accessed May 17, 2023).

[2]
“MCP23008,” *Microchip.com*, 2023. https://www.microchip.com/en-us/product/MCP23008 (accessed May 17, 2023).

[3]
Adafruit Industries, “Adafruit DS3231 Precision RTC Breakout,” *Adafruit.com*, 2016. https://www.adafruit.com/product/3013 (accessed May 17, 2023).

[4]
“Nano | Arduino Documentation,” *Arduino.cc*, 2023. https://docs.arduino.cc/hardware/nano (accessed May 17, 2023).

[5]
“DFPlayer Mini Mp3 Player - DFRobot Wiki,” *Dfrobot.com*, 2016. https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299 (accessed May 17, 2023).

Acknowledgement and Credits

I’d like to thank Nic the technician for providing components/replacements when needed, opening the workshop early in the morning and for his never-ending patience.

Appendix

A1. Bill of Materials

#	Name of Product	Part #	Quantity	Unity Cost	Cost
1	Stepper Motor (With Driver Board)	MOT-28BYJ48	2	$ 6.95	$ 13.90
2	PIC Microcontroller	PIC16F690-I/P	1	$ 4.54	$ 4.54
3	RTC Module	ARD-DS3231	1	$ 8.79	$ 8.79
4	Rotary Encoder	SW-ENC-03	1	$ 4.38	$ 4.38
5	Arduino Nano Board	ABRANANO	1	$ 19.45	$ 19.45
6	Magnetic Hall Effect Sensor	SENS-67	2	$ 2.79	$ 5.58
7	DFPlayer - MP3 Player Module	DFR0299	1	$ 15.31	$ 15.31
8	Round Magnets 3x3mm (20 Pack)	MAG-N-03	1	$ 2.41	$ 2.41
9	Speaker - 4ohms 3 Watts	1314-ADA	1	$ 3.76	$ 3.76
10	IC Socket 18 Pin	18LP	1	$ 0.15	$ 0.15

11	IC Socket 20 Pin	20LP	1	$ 0.16	$ 0.16
12	Slide Switch SPDT	SSW-110	3	$ 0.60	$ 1.80
13	Normal Open - Push Button	NO-112RD	5	$ 1.20	$ 6.00
14	40 Pin Header	FH-1	1	$ 1.07	$ 1.07
15	2-Pin Terminal Block, 4 Pack	2491P	3	$ 3.46	$ 10.38
16	3-Pin Terminal Block, 3 Pack	2492P	1	$ 3.84	$ 3.84
17	4-Pin Terminal Block, 2 Pack	2493P	2	$ 3.46	$ 6.92
18	5-Pin Terminal Block, 2 Pack	2494P	1	$ 4.23	$ 4.23
19	PCB	Y3	1	$ 18.38	$ 18.38
				Subtotal	$ 131.05
				Total	$ 150.71

Sources along with links and receipts are provided in the BOM excel sheet provided along with this document.

Most of the pieces mentioned above were obtained through ABRA Electronics.

A2. Media

Reference Number	Link	Description
1	Prototyping (Clock).jpg	Prototyping & coding the clock with the PIC 16F690.
2	Prototyping MP3.jpg	Prototyping & coding the MP3 with the Arduino and the DFPplayer.
3	PCB Board.jpg	PCB Board as delivered by JLCPCB.
4	PCB Board Populated.jpg	PCB Board with component soldered on.
5	3D Clock.png	3D Model of the clock. Model is an Assembly in Solidworks.
6	Testing with PCB.jpg	Testing the PCB Board with the faceplate.
7	Testing with PCB 2.jpg	Testing the PCB Board with the faceplate.
8	Gearbox.jpg	3D Printing and Assembling the gearbox of the clock.
9	Clock Without Faceplate.jpg	3D Printing and Assembling the clock. Without faceplate.
10	Final Product.jpg	Final Product fully assembled.
11	Proof of work.mp4	Proof of final product working.

A3. Engineering and Manufacturing Documentation

Reference Letter	Link	Description
A	Time & Alarm (PIC16F690)	Code - PIC16F690
B	MP3_Play_Arduino	Code - Arduino
C	Clock Parts & Assembly	3D Model - Parts & Assembly
D	Maya A. Rosales O._P2_Drawings_FULL.pdf	3D Model - Drawings - PDF
E	3D Printing Files	3D Model - STL Files
F	CNC Files	3D Model - DXF Files
G	PCB.dch	PCB - Schematic
H	Pattern Files	PCB - Pattern Files
I	PCB.dip	PCB - Board
J	PCB_gerberx2_Y3.zip	PCB - Gerber Files

A4. Miscellaneous

N/a