Makerere University

P.O. Box 7062, Kampala Uganda

Tel: +256-414-532634

Website: www.mak.ac.ug

Email: cis.mak.ac.ug

Embedded and Real Time Systems

Mr. Kavuuma Pius

Artificial Egg Incubator

Name	Student Number	Registration Number
Muwanga Sudaice	2100712763	21/U/12763/EVE
Mukwaya Shawn Mels	2100723354	21/U/23354/EVE
Mukisa Jotham Prince	2100708970	21/U/08970/EVE
Beheram Zena	2100713518	21/U/13518/PS
Wantante Fortune Semeon	2100712262	21/U/12262/EVE

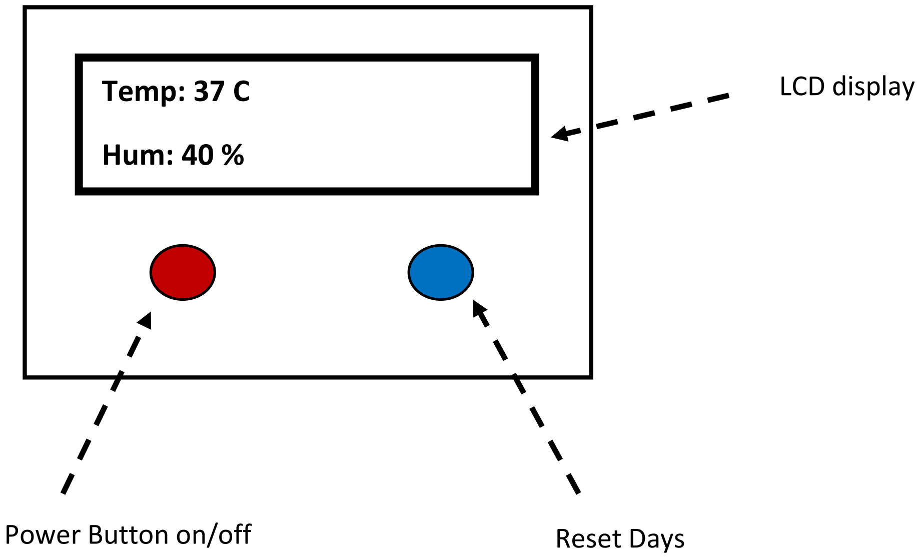
Circuit Design Link

https://www.tinkercad.com/things/k3UEV0KOCqv?sharecode=aU9M-KKBp1eX-5dykp2yqc0U0Rx2ZcEW3MZaP5SgDoA

3D Design Link
https://www.tinkercad.com/things/aQqVSRv6ktn?sharecode=Bf5Uv9haNxD4lT1-ERnHimdcb1VbENKJ61Cu-vUa640

Table of Contents

1.0 Mock Up Diagram of the User Interface for the Incubator

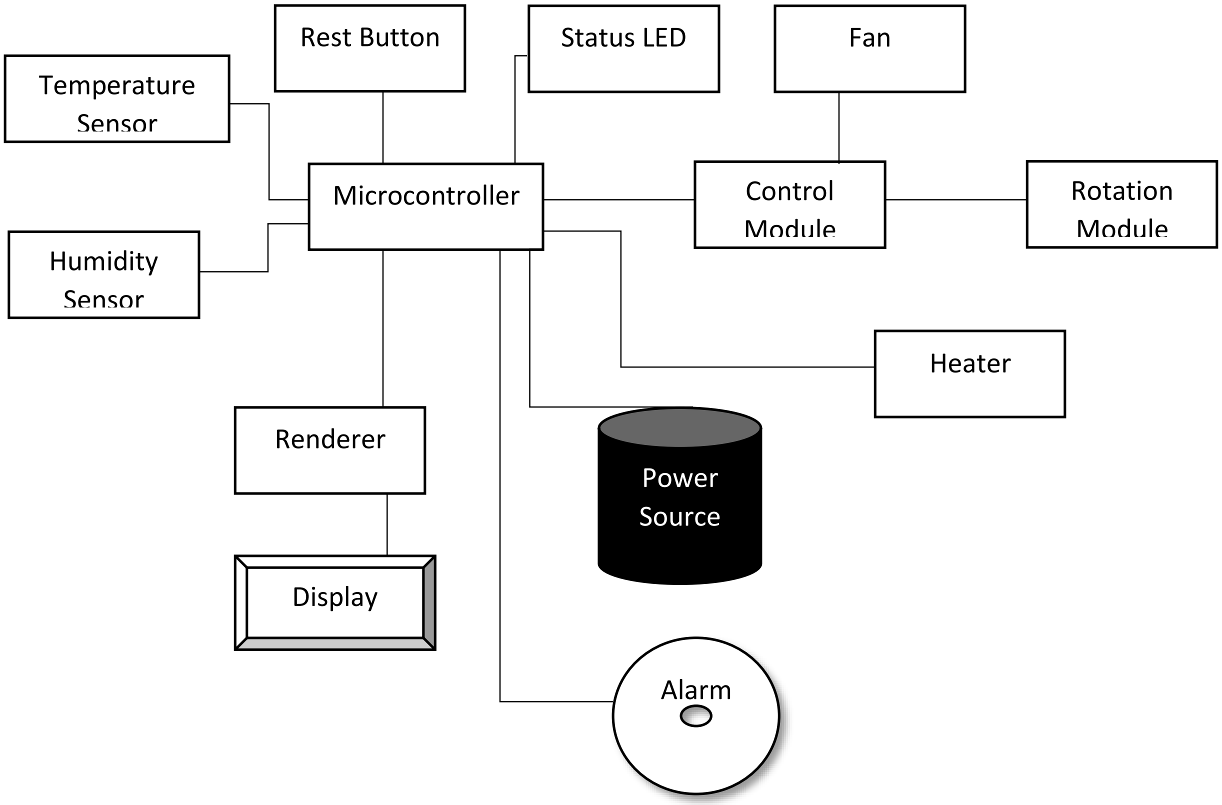


2.0 Requirements Document

NAME	CHICKEN EGG INCUBATOR
PURPOSE	Increase the hatching rate
FUNCTIONS	The incubator accommodates 192 eggs and provides the suitable and necessary temperature and humidity conditions to guarantee a high hatching rate.
INPUTS	Buttons(power and reset)
OUTPUTS	Alarm, RGB LED, Fan, Heater, LCD display panel
POWER	4.33KW
MANUFACTURING COST	UGX 300,000
PERFORMANCE	Automatic temperature and humidity control. An alarm that notifies when the days are done.
PHYSICAL WEIGHT AND SIZE	35”x24’’x26’’ 350 lbs

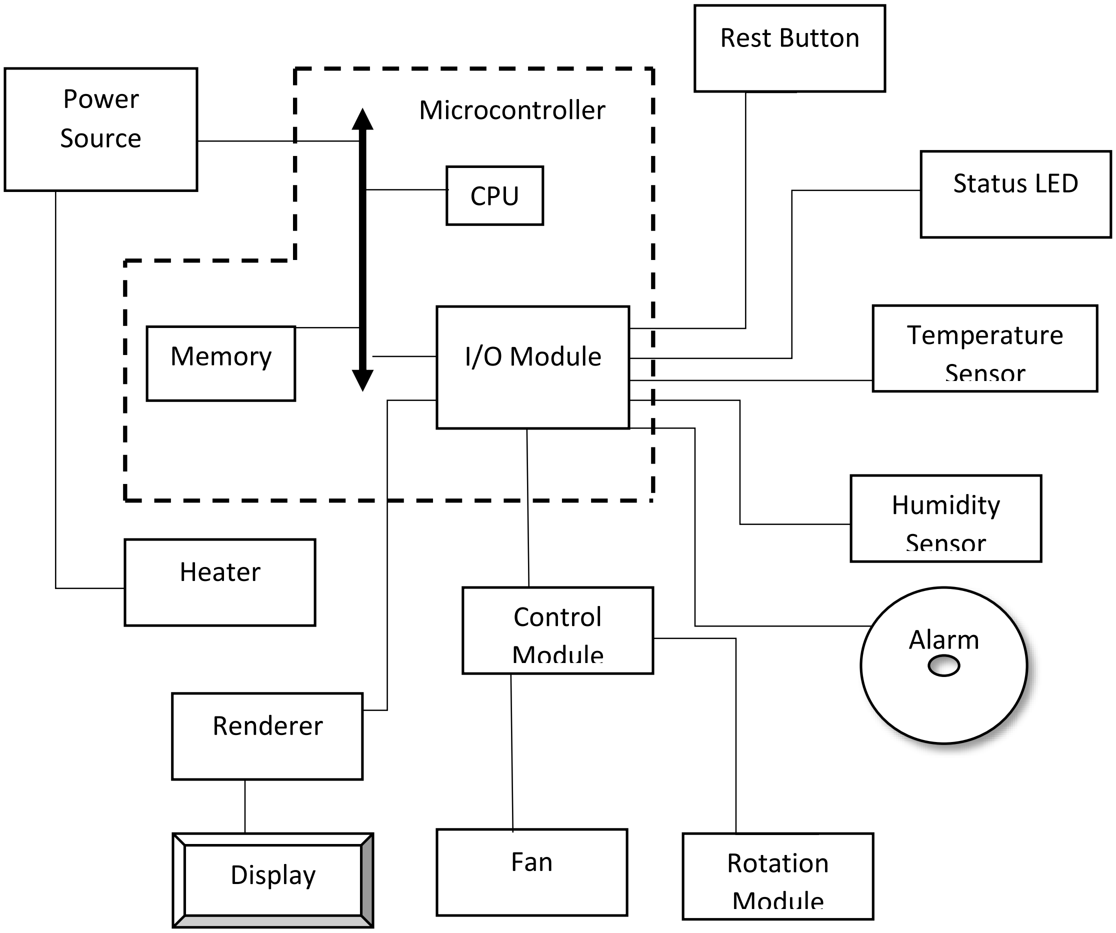
3.0 Architectural Design

3.1 Block Diagram

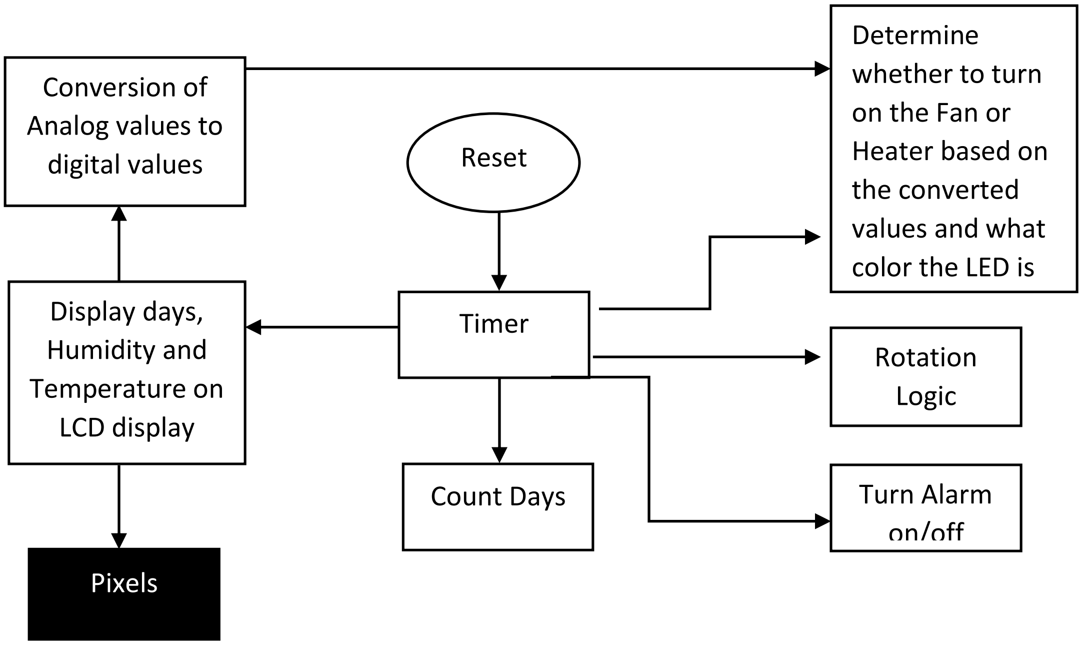


3.2 Hardware and Software Architecture

3.2.1 Hardware Architecture



3.2.2 Software Architecture


Conversion of Analog values to digital values
Reset
Determine whether to turn on the Fan or Heater based on the converted values and what color the LED is
Display days, Humidity and Temperature on LCD display
Timer
Rotation Logic
Count Days
Turn Alarm on/off
Pixels

4.0 Description

4.1 Structural Description

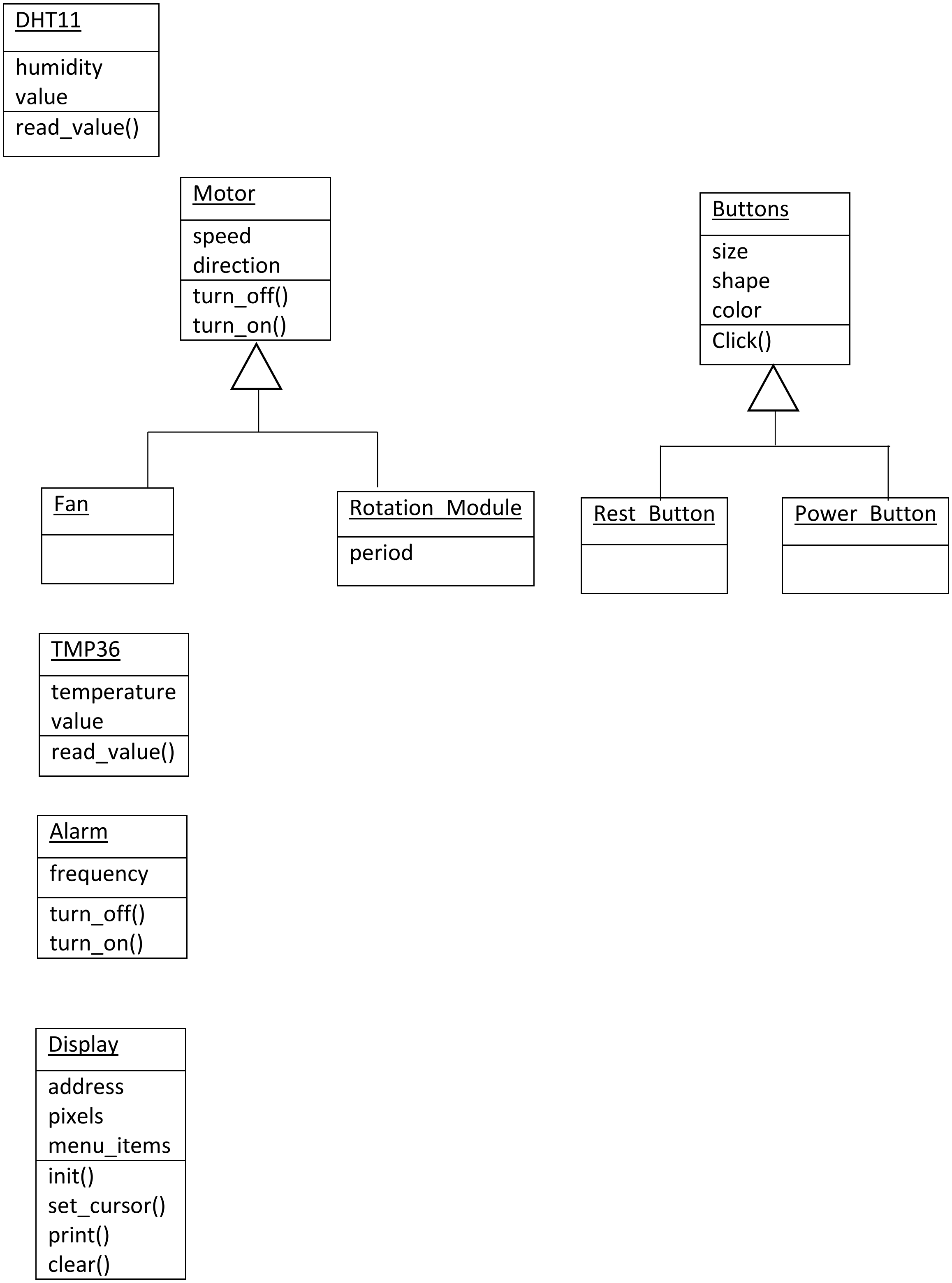
A detailed description of the objects, attributes and there methods of the Incubator

Heater
temperature

turn_off()
turn_on()

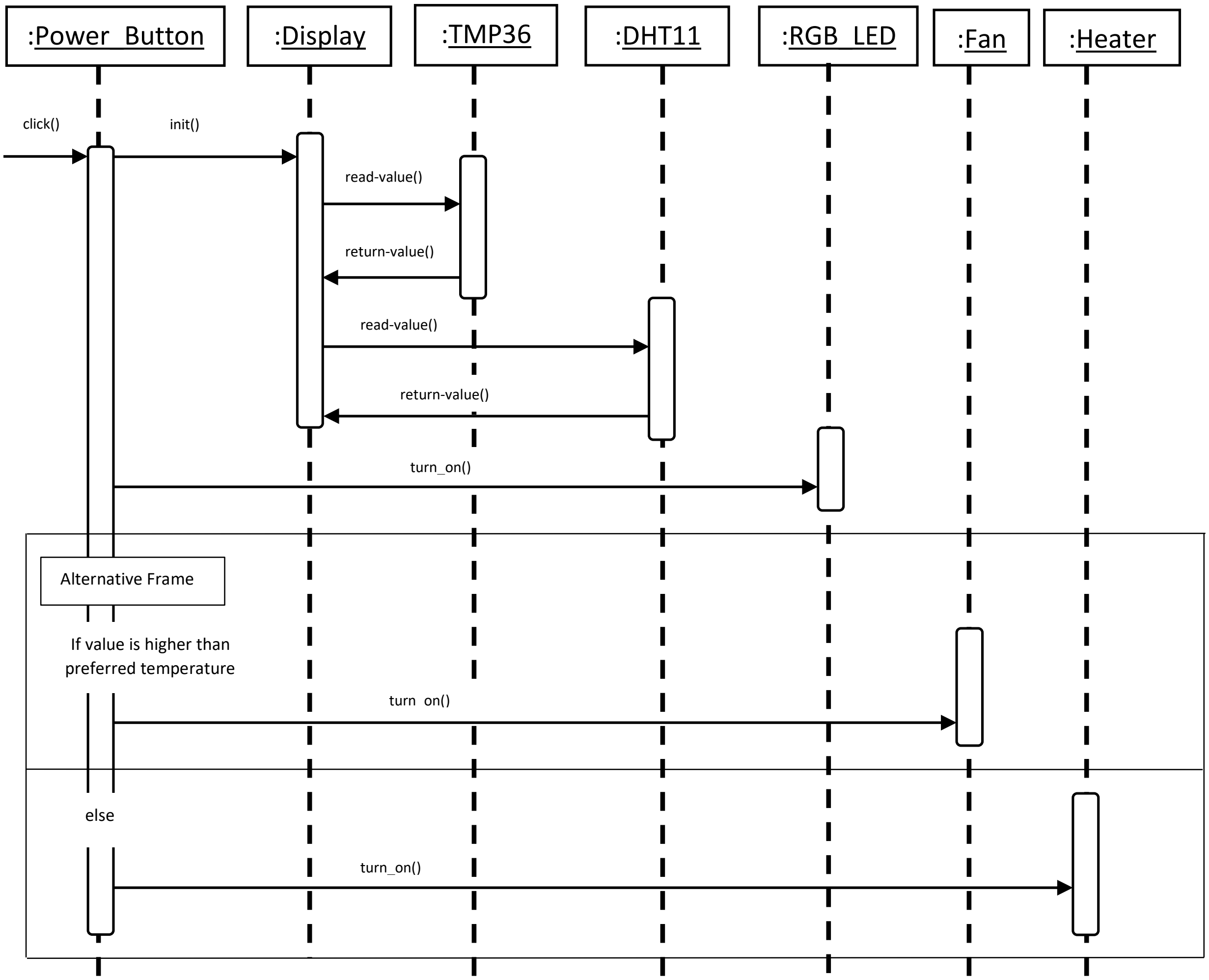
RGB_LED
color

turn_off()
turn_on()

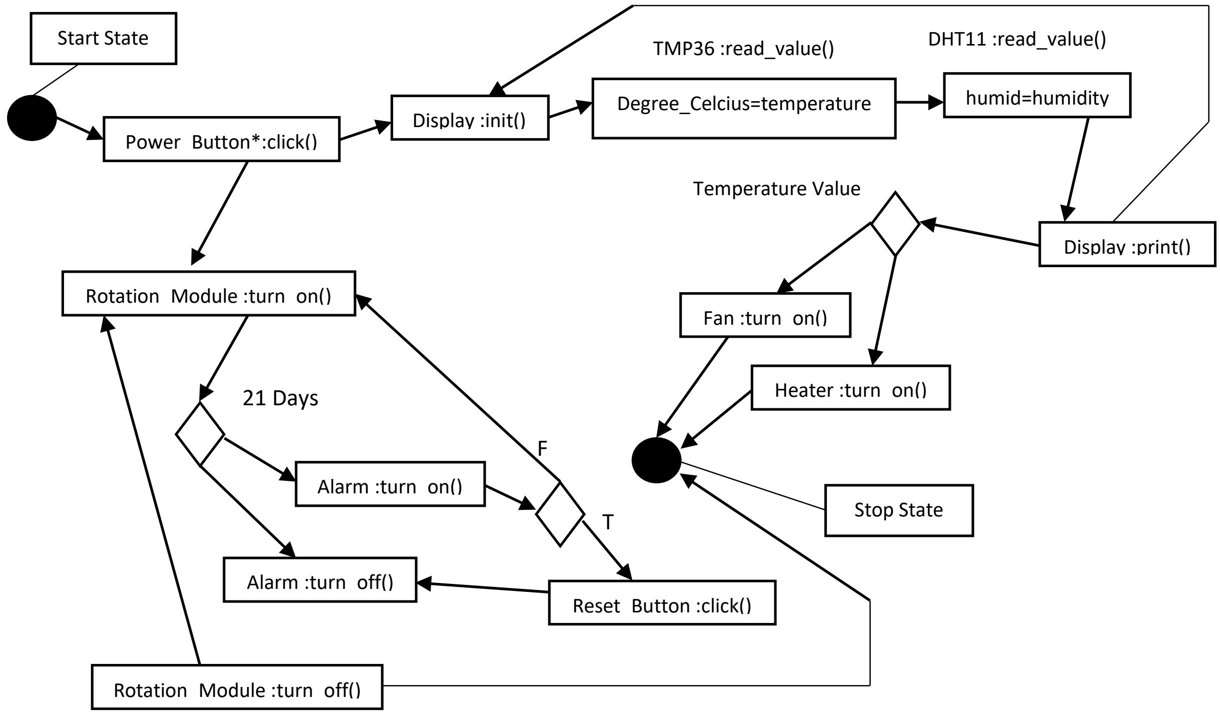

DHT11
humidity
value
read_value()
Motor
speed
direction
turn_off()
turn_on()
Buttons
size
shape
color
Click()
Fan
Rotation Module
period
Rest Button
Power Button
TMP36
temperature
value
read_value()
Alarm
frequency
turn_off()
turn_on()
Display
address
pixels
menu_items
init()
set_cursor()
print()
clear()

4.2 Behavioral Description

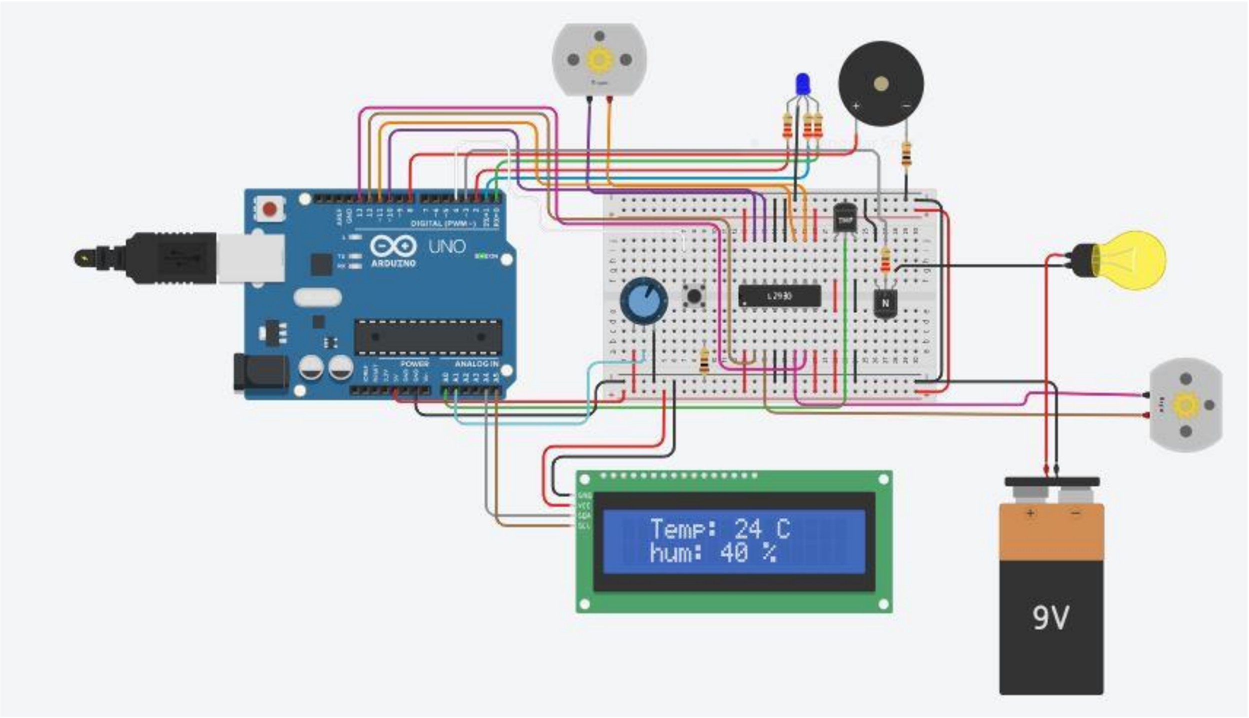
4.2.1 Sequence Diagram



4.2.2 State Diagram



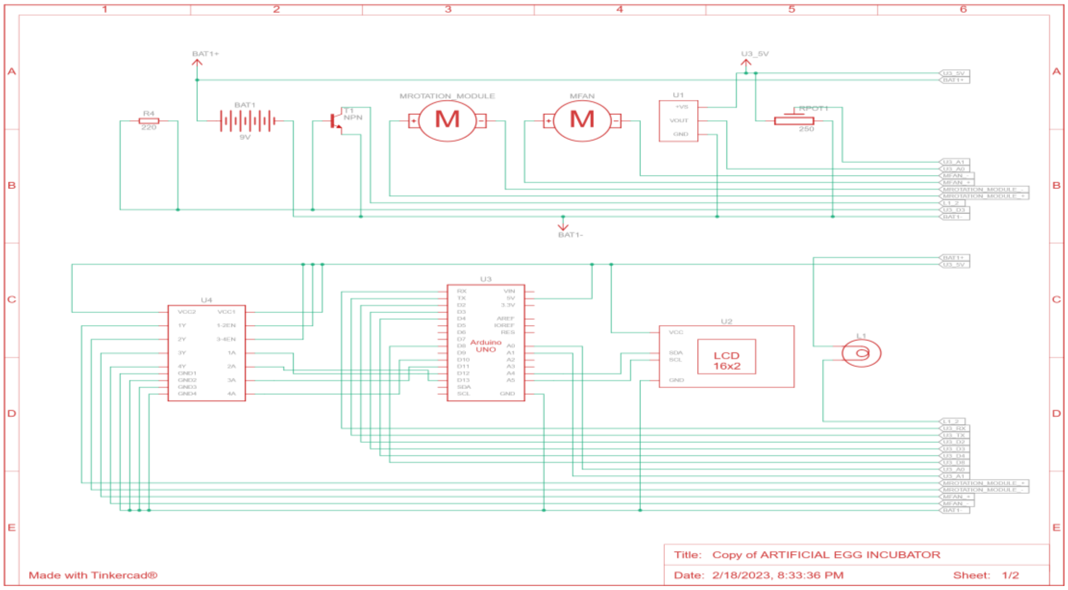
5.0 Circuit

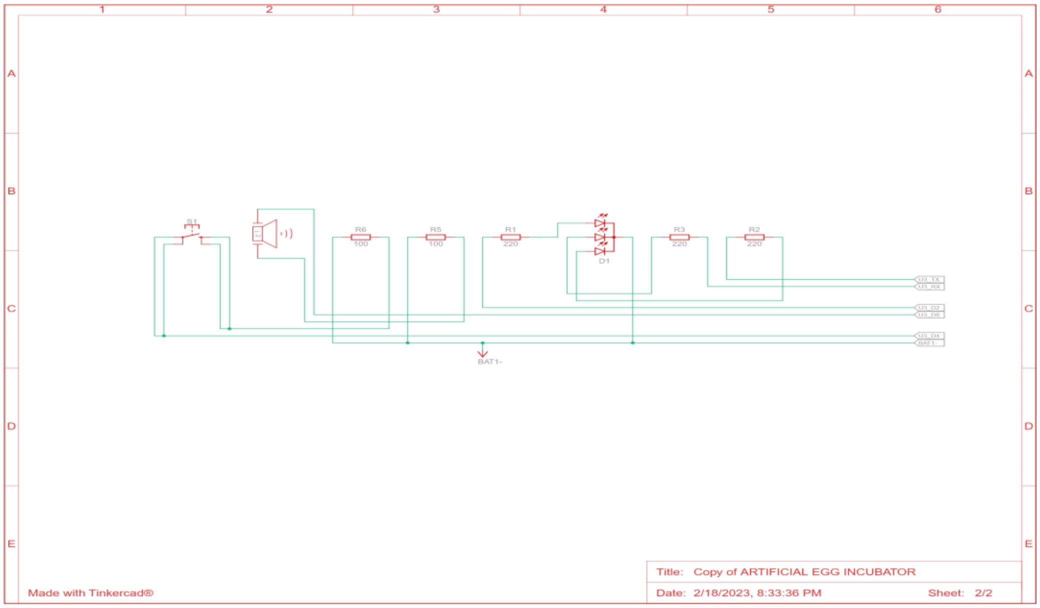


5.1 Components

Name	Quantity	Component
U3	1	Arduino Uno R3
Rport1	1	250 Ω Potentiometer
U1	1	Temperature Sensor [TMP36]
R2 **R3** **R1** **R4**	4	220 Ω Resistor
U4	1	H-bridge Motor Driver
L1	1	Light Bulb
T1	1	NPN Transistor (BJT)
MFan **MRotation Module**	2	DC Motor
BAT1	1	9V Battery
PIEZO1	1	Piezo
R5 **R6**	2	100 Ω Resistor
S1	1	Pushbutton
U2	1	PCF8574-based, 39 LCD 16 x 2 (I2C)
D1	1	LED RGB

6.0 Schematic





7.0 Code

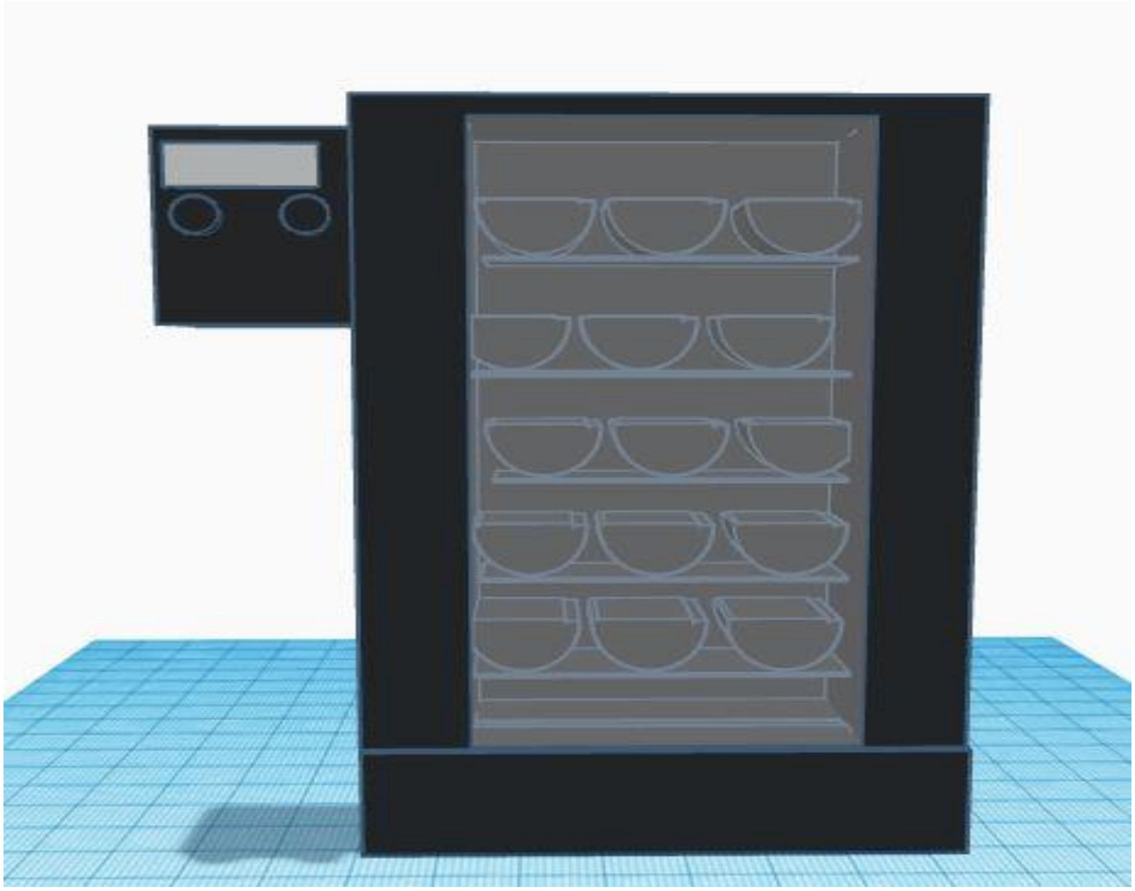
```
//TMP36 records temperatures from -40 to 125 degrees celcius

//Low value reads at 20 and high value reads at 358


#include <LiquidCrystal_I2C.h>

#include <Wire.h>


LiquidCrystal_I2C lcd(0x27, 16, 2);

volatile uint8_t days = 1;

volatile uint8_t rotations = 0;

volatile bool interrupt_occured = false;

bool firstTime = true;

bool LCDfirstTime = true;

uint16_t temp = 0;
```

```
uint16_t humid = 0;
uint8_t theLow;
uint16_t theTenBitResult;
int degrees_celcuis;
short percentage_humidity;
volatile static uint8_t adc_convert_done = 1;

void Alarm(){
  int cycles = 400;
  int counter = 0;
  while(counter < cycles){
    int i;
    PORTB |= 0x01;
    for(i=0; i<512; i++);
    PORTB &= ~(0x01);
    for(i=0; i<512; i++);
    counter++;
  }
}
void FirstTime(){
  PORTB |= 0x20;
  CustomDelay(250);
  PORTB &= ~(0x20);
  firstTime = false;
  rotations++;
}
void SecondTime(){
  PORTB |= 0x10;
  CustomDelay(250);
  PORTB &= ~(0x10);
  firstTime = true;
  rotations++;
}

void Rotations(){
  if(rotations < 5){
     Rotator();
     return;
  }
  days++;
  rotations = 0;
}

void Rotator(){
  if(firstTime){
    FirstTime();
    return;
  }
  else {
    SecondTime();
    return;
  }
}

void LCD_Display(){
  if(LCDfirstTime){
    lcd.setCursor(1, 0);
    lcd.print("Days ");
    lcd.print(days);
    CustomDelay(200);
    lcd.clear();
    LCDfirstTime = !LCDfirstTime;
    return;
  }else{
    lcd.setCursor(2, 0);
    lcd.print("Temp: ");
            lcd.print(degrees_celcuis);
            lcd.print(" C");
    lcd.setCursor(2, 1);
            lcd.print("hum: ");
    lcd.print(percentage_humidity);
    lcd.print(" %");
            CustomDelay(200);
            lcd.clear();
    LCDfirstTime = !LCDfirstTime;
    return;
  }
}

void CustomDelay(uint32_t mSecondsApx)
{
  volatile uint32_t long i;
  uint32_t endTime = 1000 * mSecondsApx;
  for (i = 0; i < endTime; i++);
}

ISR(TIMER1_COMPA_vect){
  interrupt_occured = true;
}

ISR(ADC_vect){
  adc_convert_done = 1;
}

ISR(PCINT2_vect){
  //Runs when pin change interrupt 2 occurs
  days = 1;
  rotations = 0;
}
void setup()
{
  //The setup code
  //inputs
  PORTD |= 0x10; //enable internal pull up resistor on pin 4
  ADMUX = 0x40; //select A0 and AVcc voltage reference selection
  ADCSRA = 0x8F; //enable ADC, ADC interrrupts and 128 prescaler
  DIDR0 = 0x03; //enable ADC 0 and 1
  //outputs
  DDRD |= 0x0F; //set only the RGB LED and Piezo as output
  DDRB |= 0x3D;
  //Pin change interrupt initialization
  PCICR = 0x04; //enable PCINT[23:16] pin interrupts
  PCMSK2 = 0x10; //enable pin change interrupt on pin 4
  //Timer 1 initialization
  TCCR1A = 0;
  TCCR1B = 0x0D; // CTC mode with prescaler 1024
  OCR1A = 31249; //interrupt period 2s
  TIMSK1 = 0x02; //enable interrupt on compare match 1
  sei(); //enable global interrupts
  //LCD initialization
  lcd.init();
  lcd.backlight();
}

void loop()
{
  //the loop
  LCD_Display();
  adc_convert_done = 0;
  ADCSRA |= 0x40; //To start the conversion
  while(adc_convert_done == 0);
  theLow = ADCL;
  theTenBitResult = ADCH<<8|theLow;

   switch(ADMUX){
    case 0x40:
        temp = theTenBitResult;
        ADMUX = 0x41;
        break;
    case 0x41:
        humid = theTenBitResult;
        ADMUX = 0x40;
        break;
    default:
        break;
```

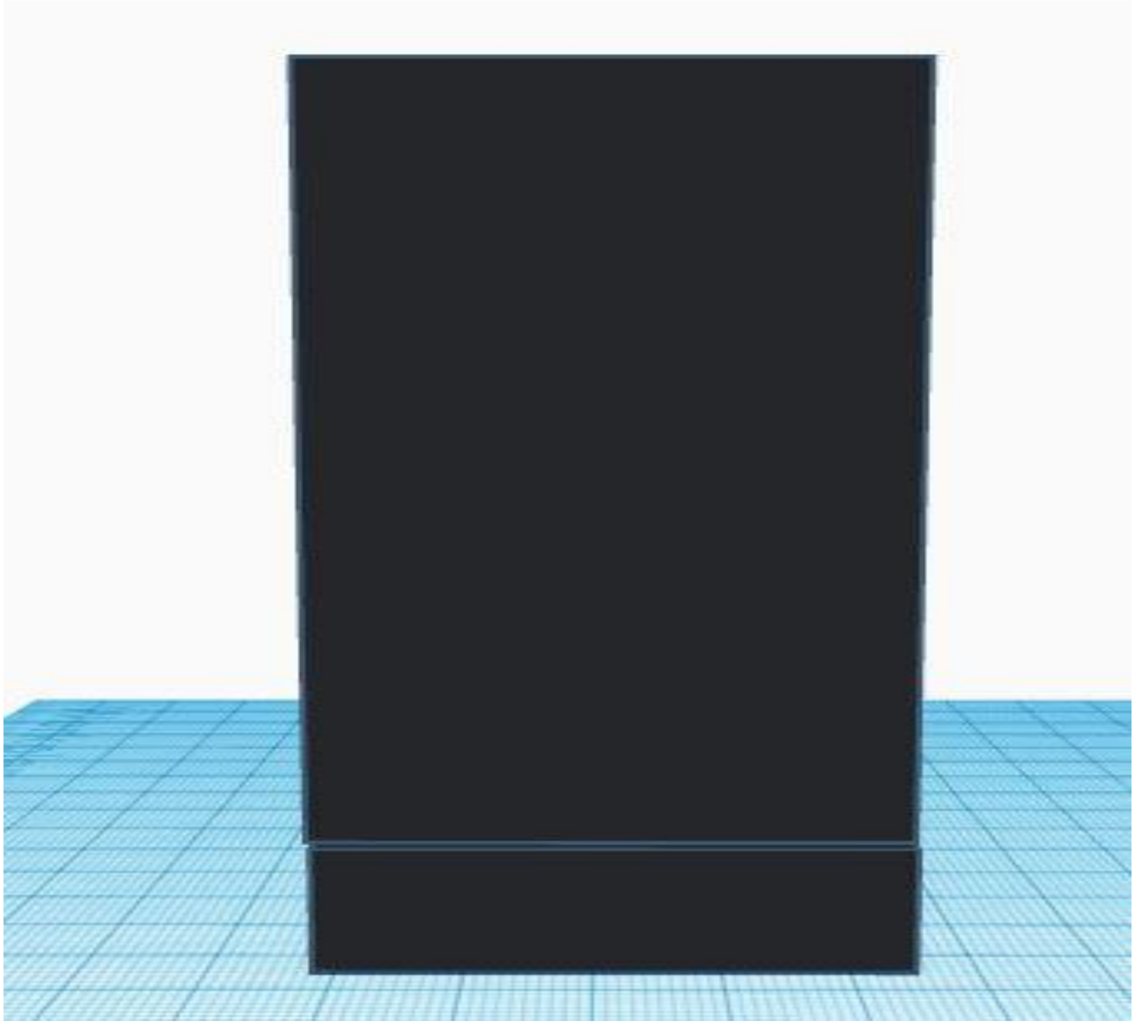
```
}

/* To perform some magic on the temp in volts
in order to display it in degrees Celcius */
degrees_celcuis = ((temp*(5.0/1024.0))-0.5)/0.01;

/* To perform some magic on the humidity in volts
in order to display it as a percentage */
percentage_humidity = (humid*100)/1024;

//Temperature variation logic
switch(degrees_celcuis){
  case -40 ... 35:
        PORTD &= ~(0x05); //clear red and green
        PORTD |= 0x0A; // set blue and turn on heater
        PORTB &= ~(0x04); //turn off the Fan
        break;
  case 36 ... 40:
        PORTD &= ~(0x0E); //clear red and blue and turn off heater
        PORTD |= 0x01; //set green
        PORTB &= ~(0x04); //turn off the Fan
        break;
  case 41 ... 125:
        PORTD &= ~(0x0B); //clear blue and green and turn off heater
        PORTD |= 0x04; //set red
        PORTB |= 0x04; //turn on the Fan
        break;
  default:
        //pass
        break;
    }
    if(interrupt_occured){
      Rotations();
      interrupt_occured = false;
    }
    if(days >= 21){
      Alarm();
    }
}
```

8.0 3D design

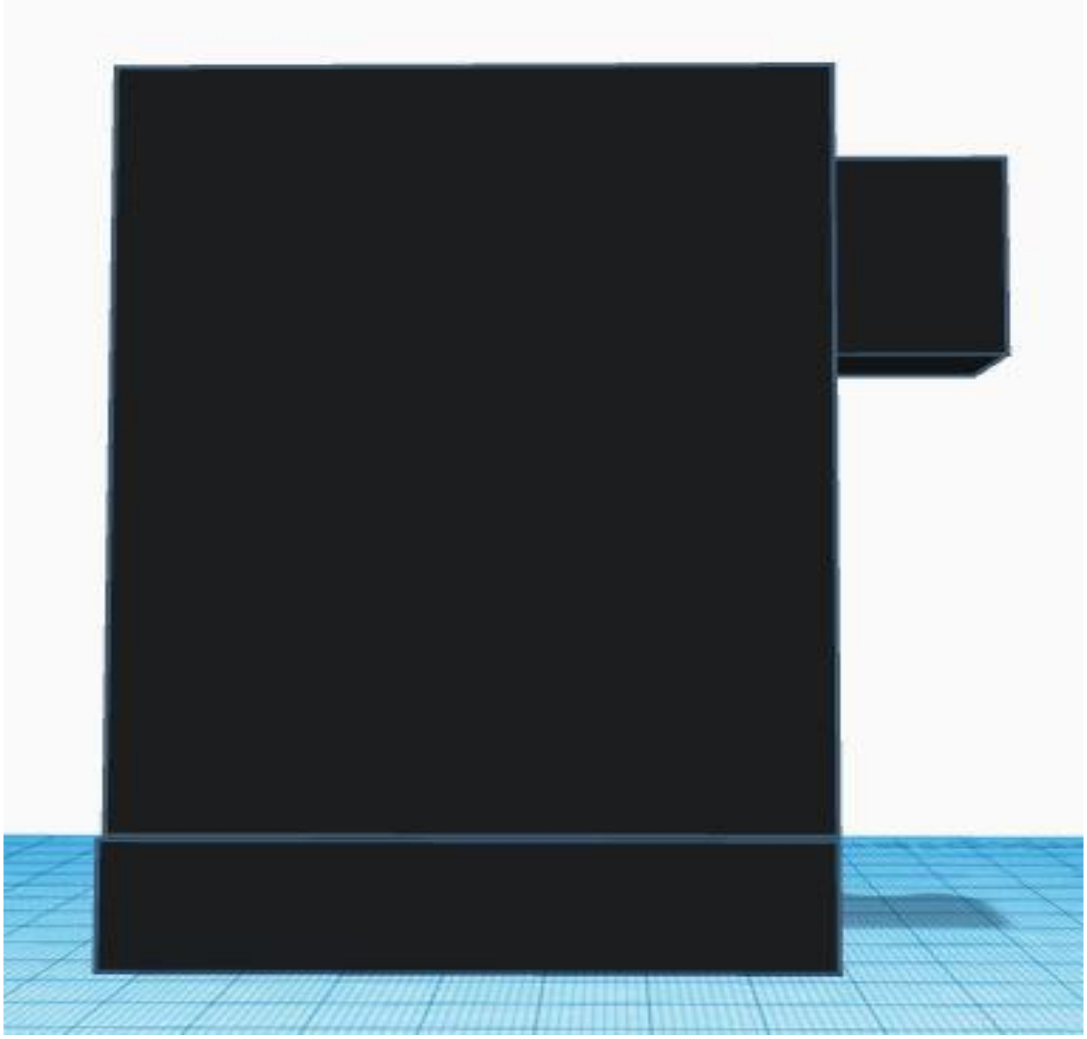
8.1 Front



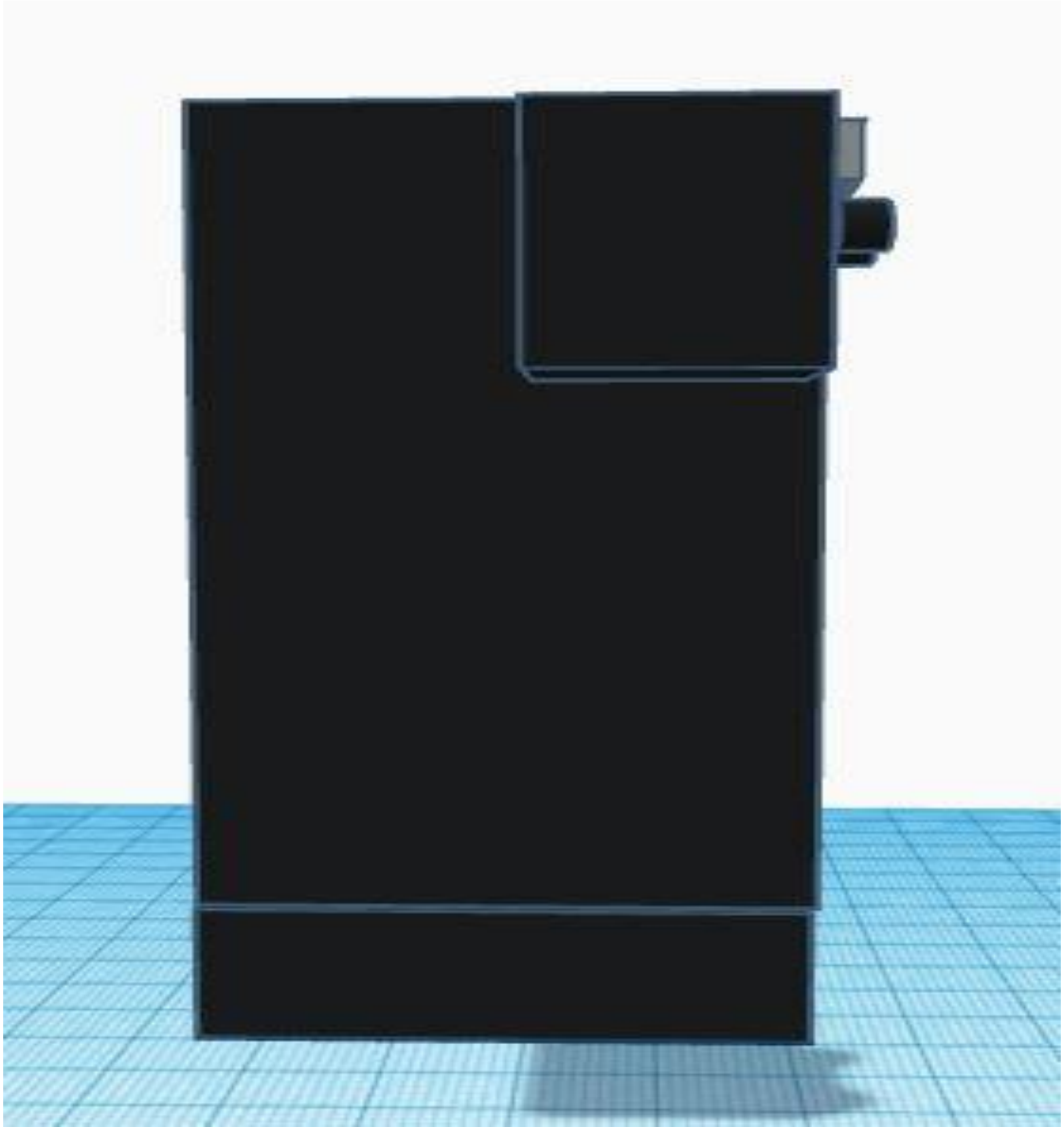
8.2 Right



8.3 Back



8.4 Left



9.0 Conclusion and Remarks

I. Given the fact that there are several LCD types and brands that we could us in the project (PCF8574-based and MCP23008-based) there was no specific way to control the LCD using registers from scratch because every LCD model is built different and the manufacturers provide a specific library that is supposed to interact with their specific LCD.
Replicating the code in the library is very complex and requires a lot of information about how the LCD was built and this is why we used the LiquidCrystal_I2C.h for the PCF8574-based LCD to communicate with it.

II. We used a potentiometer as our humidity sensor because the DHT11 was removed from Tinkercad.

When you vary the resistance of the potentiometer, the current at the analog input pin A1 changes and we use the amount of current at the pin to indicate the level of humidity as a percentage.

III. We use a bulb as our heater because of the absence of a heater in the hardware components in Tinkercad.

IV. We have 2 Servo DC motors controlled via an H-bridge motor chip. One motor is used to indicate the rotation of the fan and the other rotation module rotates eggs periodically so as to prevent the embryo of the chick to stick on one side of the egg.

A day is counted after 5 rotations of the rotation module that happens every after 2 seconds indicating 4.5 hours in real-time that the eggs are supposed to turn.

V. The push button is used to reset the days back to day 1 of incubation i.e. that is after 21 days.

VI. The Piezo alarm goes off to alert farmers to remove the hatched eggs from the incubator after 21 days in which they are expected to have hatched.