ENGR-UH 3110

Instrumentation, Sensors, Actuators

Multi-Stage TEC-based Compressor-Less Refrigerator

Project Report

Week 3 - Prototyping The Control System & Sensor Testing

Group II

Omar Al Aleeli [oa2045]

Ahmed Alkindi [aaa9888]

Hiba Assamaouat [ha1647]

Thea Hayek [th2746]

Muhammad Ibrahim [mi2171]

Kundai Mutuwira [km5389]

Marine Ramaroson [mmr9867]

Instructor

Muhammad Faraz Sheikh

جامعة نيويورك أبوظبي

NYU | ABU DHABI

Table of Contents

Note: The GitHub Repository for this project is accessible via this link.

Objectives:

- Set up the Arduino with the DS18B20 sensor(s) and LCD.
- Develop and test basic code to read the temperature and display it.
- Validate sensor calibration and wiring integrity on a breadboard.

Temperature measurement:

Components:

- Arduino Uno R3
- DS18B20 Temperature Sensor
- 52Pi I2C OLED Display (0.96" 128x64px)
- Breadboard
- Connection Wires
- 4.7 kΩ Resistor (for pull-up resistor)

The diagram below shows a sample wiring diagram used in setting up the circuit using an LM35 temperature sensor:

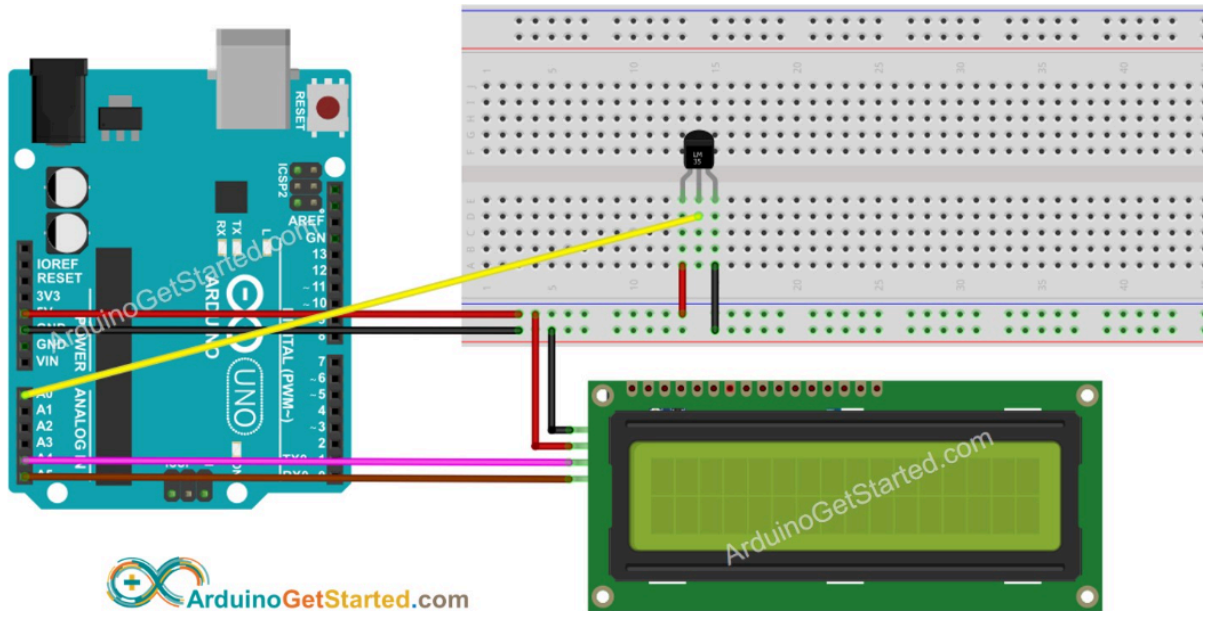


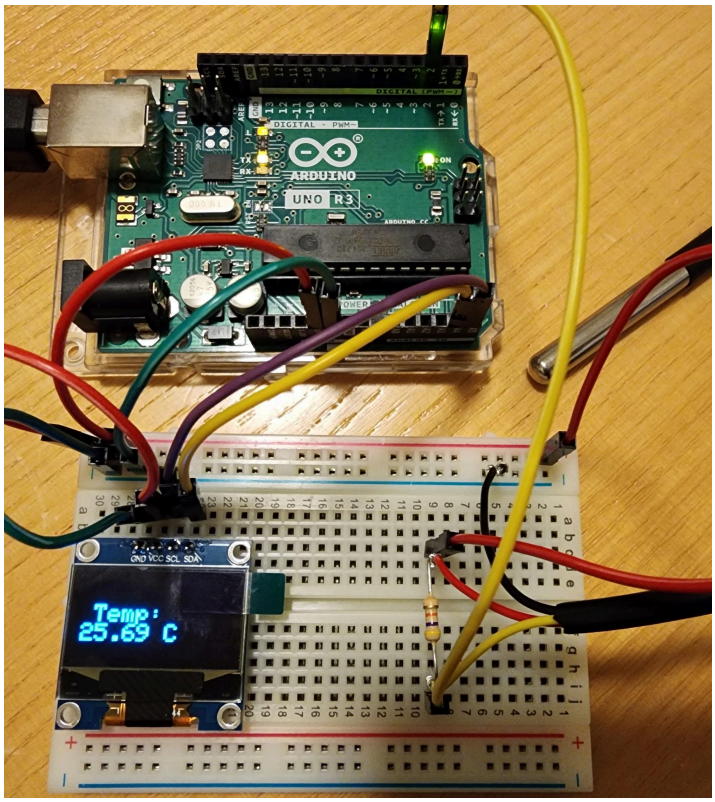
Figure 1: Sample Wiring diagram.

The OLED display (I2C) is connected to the Arduino Uno R3 using the following pins:

OLED display	Arduino Uno R3
VCC	5V
GND	GND
SDA (data signal)	A4
SCL (clock signal)	A5

DS18B20 Temperature Sensor is connected to the Arduino Uno R3 using the following pins:

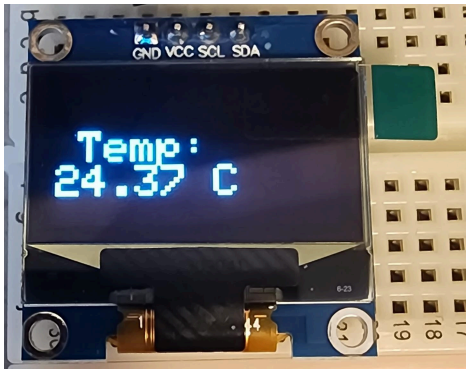
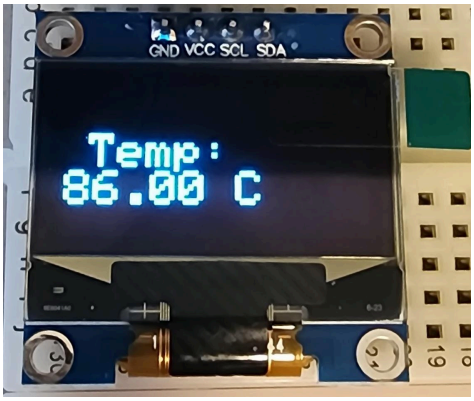
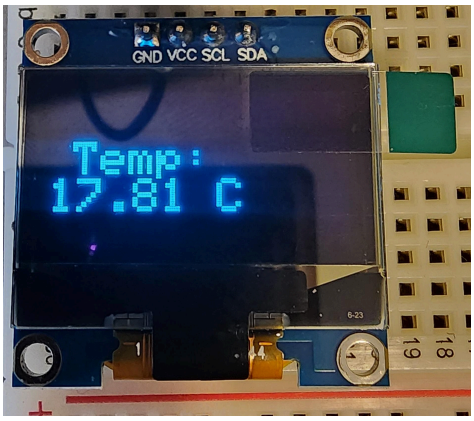
DS18B20 Sensor	Arduino Uno R3
VCC (Red)	5V
GND (Black)	GND
Data signal (yellow)	Digital pin 2



The 4.7 kΩ Resistor is connected between the DS18B20 Sensor's data line and 5V. The DS18B20 communicates using the OneWire protocol, which requires the data line to be pulled high when idle.

The figure 2 to the left, shows the circuit designed to measure room temperature using the DS18B20 temperature sensor while displaying the room temperature on the OLED display.

The table below shows different temperature measurements done using the temperature sensor:

System	Measurement from DS18B20 Sensor
Room Temperature Set at 24.5 ℃	
Boiling Water	
Cold Tap Water	

Source Code:

Figures below show the code used in recording the temperature using figure 2:

```
#include <Wire.h>
#include <Adafruit_GFX.h> // Library for the LCD display
#include <Adafruit_SSD1306.h> // Library for the LCD display
#include <OneWire.h> // Library for the DS18B20 sensor
#include <DallasTemperature.h> // Library for the DS18B20 sensor

// OLED Display Settings
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// DS18B20 Sensor Settings
#define ONE_WIRE_BUS 2  // DS18B20 data pin connected to D2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup() {
    Serial.begin(9600);

    // Initialize OLED
    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println(F("SSD1306 allocation failed"));
        for (;;);
    }
    sensors.begin(); // Initialize DS18B20 Sensor
    display.clearDisplay(); // Clear OLED display
}

void loop() {
    sensors.requestTemperatures();  // Request temperature from sensor
    float temperatureC = sensors.getTempCByIndex(0);  // Get temperature in Celsius

    //Print to Serial Monitor
    Serial.print("Temperature: ");
    Serial.print(temperatureC);
    Serial.println("°C");

    // Display temperature on OLED
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(10, 20);
    display.print("Temp: ");
    display.print("   ");
    display.print(temperatureC);
    display.print(" C");
```

```
    display.display();

    delay(1000);  // Update every 1 second
}
```

Fun Code:

Chilly McFrosty is a reaction-based game designed to test the player's response time in comparison to an average F1 driver. The game uses an OLED display, an LED system, and a push button to create an engaging experience. The player must press the button as quickly as possible after receiving a visual cue. If their reaction time is fast enough, they win; otherwise, "Tough Luck".

Components:

- Arduino Uno R3
- 52Pi I2C OLED Display (0.96" 128x64px)
- Breadboard
- Green LED (Success Indicator on pin 6)
- Red LED (Failure Indicator on pin 5)
- Push Button (Connected to pin 8 with internal pull-up)
- 10 kΩ Resistor
- Connection Wires

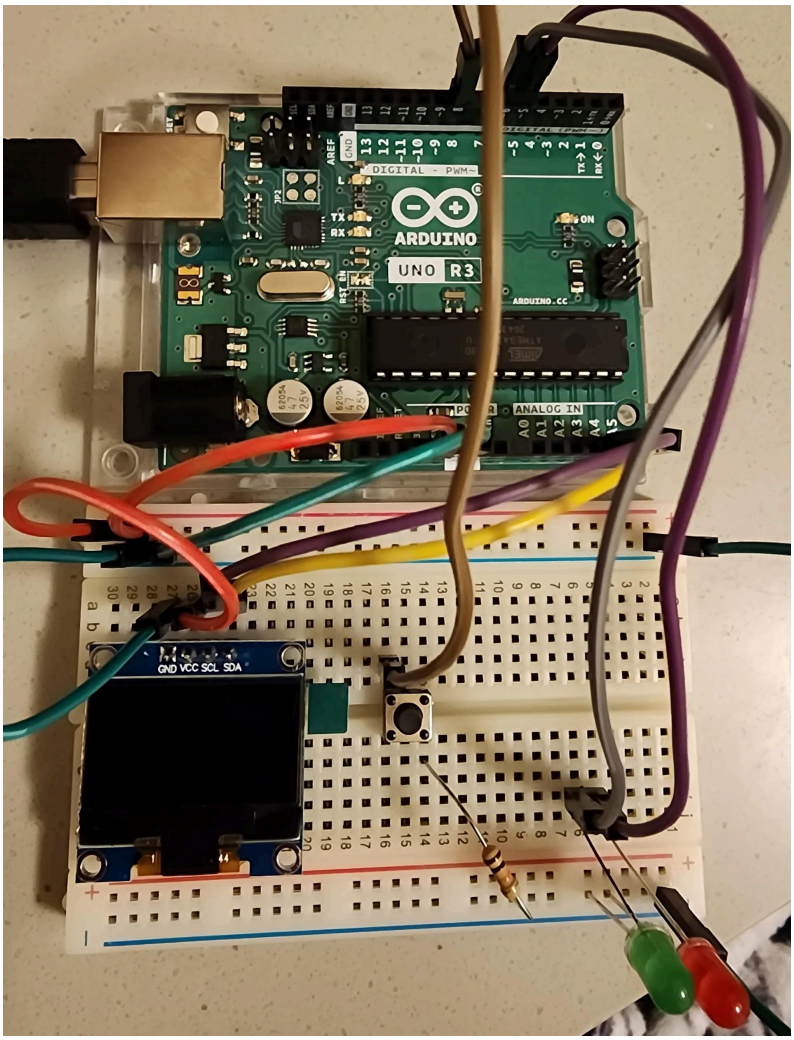


Figure 3: Reaction time game circuit setup.

OLED Game Screen:

Source Code:

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// OLED Display Setup
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// Pins
#define BUTTON_PIN 8  // Player's Button
```

```
#define GREEN_LED 6   // Success Indicator
#define RED_LED 5     // Failure Indicator

// Animation Variables
#define NUMFLAKES     10 // Number of snowflakes in the animation example
#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
#define XPOS   0 // Indexes into the 'icons' array in function below
#define YPOS   1
#define DELTAY 2
#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16

static const unsigned char PROGMEM logo_bmp[] =
{ 0b00000000, 0b11000000,
  0b00000001, 0b11000000,
  0b00000001, 0b11000000,
  0b00000011, 0b11100000,
  0b11110011, 0b11100000,
  0b11111110, 0b11111000,
  0b01111110, 0b11111111,
  0b00110011, 0b10011111,
  0b00011111, 0b11111100,
  0b00001101, 0b01110000,
  0b00011011, 0b10100000,
  0b00111111, 0b11100000,
  0b00111111, 0b11110000,
  0b01111100, 0b11110000,
  0b01110000, 0b01110000,
  0b00000000, 0b00110000 };

// Game Variables
#define F1_THRESHOLD 500  // Reaction Time Threshold
bool gameActive = false;
unsigned long startTime, reactionTime;

void setup() {
    Serial.begin(9600);

    // Pin Modes
    pinMode(BUTTON_PIN, INPUT_PULLUP);
    pinMode(GREEN_LED, OUTPUT);
    pinMode(RED_LED, OUTPUT);
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(RED_LED, LOW);

    // Initialize OLED Display
    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println(F("SSD1306 allocation failed"));
        for (;;);
```

```
    }
    delay(2000);
    testfillrect();

    // Welcome message
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(10, 10);
    display.setTextColor(SSD1306_WHITE);
    welcometext();
    display.display();
    delay(2000);

    testfillroundrect();
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(10, 25);
    display.println("Chilly   McFrosty!!");
    display.display();
    delay(2000);
}

// The main reaction game loop
void loop() {
    gameActive = false;

    // Display challenge message
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 0);
    display.println("Can you   beat an F1 Driver?   (200ms)");
    display.display();
    delay(2000);

    // Countdown before start
    for (int i = 3; i > 0; i--) {
        display.clearDisplay();
        display.setTextSize(5);
        display.setCursor(50, 20);
        display.print(i);
        display.display();
        delay(1000);
    }

    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(10, 20);
    display.println("Wait for  it...");
    display.display();
```

```
    delay(random(0, 5000));

    // Game starts
    display.clearDisplay();
    display.setTextSize(4);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(30, 20);
    display.println("GO!");
    display.display();
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(RED_LED, LOW);

    gameActive = true;
    startTime = millis();

    // Wait for player input
    while (gameActive) {
        if (digitalRead(BUTTON_PIN) == LOW) {
            reactionTime = millis() - startTime;
            announceResult(reactionTime);
            testanimate(logo_bmp, LOGO_WIDTH, LOGO_HEIGHT); // Animate bitmaps
            return;
        }
    }
}

// Determines the result and displays it
void announceResult(unsigned long reactionTime) {
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(10, 10);

    display.print("Time:    ");
    display.print(reactionTime);
    display.println("ms");
    display.println("         ");

    if (reactionTime < F1_THRESHOLD) {
        display.println("F1 Level  Reflexes!!");
        digitalWrite(GREEN_LED, HIGH);
    } else {
        display.println("Tough    Luck!");
        digitalWrite(RED_LED, HIGH);
    }

    display.display();
    delay(3000);
}
```

```
void testanimate(const uint8_t *bitmap, uint8_t w, uint8_t h) {
  int8_t f, icons[NUMFLAKES][3];

  // Initialize 'snowflake' positions
  for(f=0; f< NUMFLAKES; f++) {
    icons[f][XPOS]   = random(1 - LOGO_WIDTH, display.width());
    icons[f][YPOS]   = -LOGO_HEIGHT;
    icons[f][DELTAY] = random(1, 6);
    Serial.print(F("x: "));
    Serial.print(icons[f][XPOS], DEC);
    Serial.print(F(" y: "));
    Serial.print(icons[f][YPOS], DEC);
    Serial.print(F(" dy: "));
    Serial.println(icons[f][DELTAY], DEC);
  }

  for(;;) { // Loop forever...
    display.clearDisplay(); // Clear the display buffer

    // Draw each snowflake:
    for(f=0; f< NUMFLAKES; f++) {
.     display.drawBitmap(icons[f][XPOS], icons[f][YPOS], bitmap, w, h, SSD1306_WHITE);
    }

    display.display(); // Show the display buffer on the screen
    delay(200);        // Pause for 1/10 second

    // Then update coordinates of each flake...
    for(f=0; f< NUMFLAKES; f++) {
      icons[f][YPOS] += icons[f][DELTAY];
      // If snowflake is off the bottom of the screen...
      if (icons[f][YPOS] >= display.height()) {
        // Reinitialize to a random position, just off the top
        icons[f][XPOS]   = random(1 - LOGO_WIDTH, display.width());
        icons[f][YPOS]   = -LOGO_HEIGHT;
        icons[f][DELTAY] = random(1, 2);
      }
    }
  }
}

void welcometext(void) {
  display.clearDisplay();
  display.setTextSize(2); // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(10, 0);
  display.println(F("Welcome  To ..."));
  display.display();
  delay(100);
  display.startscrolldiagright(0x00, 0x07);
```

```
  delay(2000);
  display.startscrolldiagleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
}

void testfillrect(void) {
  display.clearDisplay();
  for(int16_t i=0; i<display.height()/2; i+=3) {
    // The INVERSE color is used so rectangles alternate white/black
    display.fillRect(i, i, display.width()-i*2, display.height()-i*2, SSD1306_INVERSE);
    display.display(); // Update screen with each newly-drawn rectangle
    delay(1);
  }
  delay(2000);
}

void testfillroundrect(void) {
  display.clearDisplay();
  for(int16_t i=0; i<display.height()/2-2; i+=2) {
    // The INVERSE color is used so round-rects alternate white/black
    display.fillRoundRect(i, i, display.width()-2*i, display.height()-2*i,
      display.height()/4, SSD1306_INVERSE);
    display.display();
    delay(1);
  }
  delay(2000);
}
```

Fridge model:

The refrigeration unit for this project utilizes a commercial cooler box modified to function effectively as a compact refrigerator. The original design of the cooler box features an upward-opening lid, which poses practicality limitations for standard refrigeration use. To address this, the cooler box will be reoriented onto its side, thereby converting the existing lid into a vertical door similar to conventional refrigerators.

This adaptation requires the installation of durable hinges onto the original lid, ensuring reliable operation and ease of access. The original top surface of the cooler box, now oriented to the side, will be precisely cut out to accommodate a custom-designed bracket intended to securely house the Peltier cooling modules. These modules are strategically positioned at the top of the unit to

optimize cooling efficiency, as cold air naturally descends within the enclosed space, enhancing overall temperature distribution and ensuring uniform cooling.

Additionally, sturdy legs will be installed on the bottom of the cooler box to provide stable support, lift the unit off the ground, and enhance air circulation beneath it. To manage condensation and ensure cleanliness, the existing tap originally positioned for drainage purposes will now be conveniently located at the bottom of the modified cooler. This tap allows for the easy removal of condensed water, maintaining optimal conditions within the refrigerator and simplifying ongoing maintenance.

An enclosure box will also be mounted securely on the back of the refrigerator, housing all necessary electronic components. This placement provides protection for the electronics and ensures easy access for maintenance and troubleshooting.

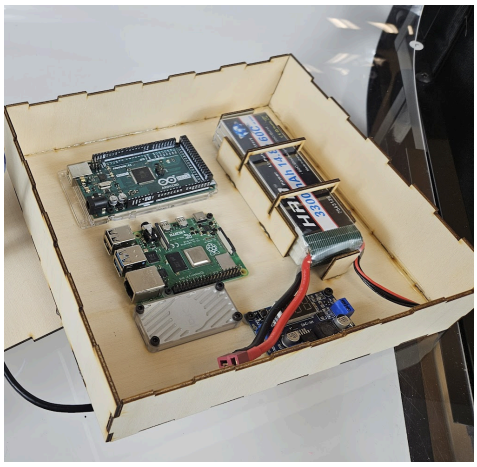


Figure 4: Enclosure box (Ignore the google coral, arduino and power supply this will be changed to our components)

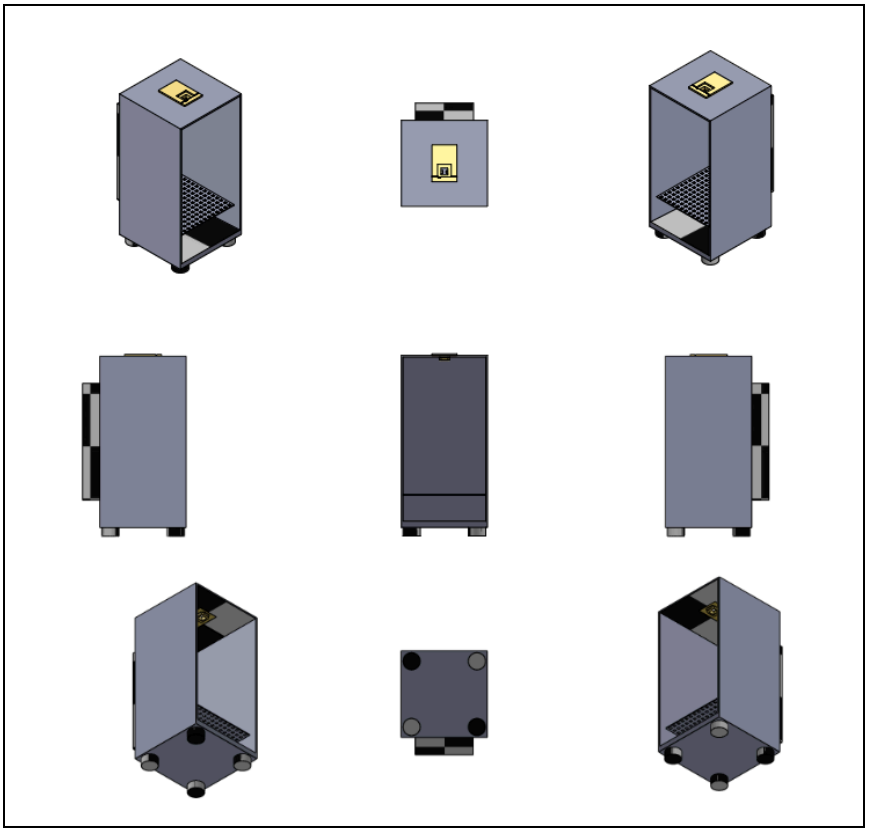


Figure 5: Refrigeration unit as per our description

References

[1] ArduinoGetStarted, "Arduino - Display Temperature from LM35 Sensor on LCD," https://arduinogetstarted.com/tutorials/arduino-display-temperature-from-lm35-sensor-on-lcd.