KING'S College LONDON

Mechatronics Project

King's College London

Library Mechatronic System

Authors: *Alexander Melarti, Alexis Leclercq, Iulia Nazarov, Sammy Noel, Javier Navarro*

January 2023

Contents

1 Introduction

Nowadays, libraries are heavily frequented not only by students, but also by people who find them useful to work or read. Because of this, it is essential that they maintain the appropriate conditions so that they can be well used. Those conditions include;

- Ensuring that the number of people in the library does not exceed the limit.
- Maintaining the right temperature inside the library.
- Making sure that the library has the appropriate humidity values so that books are not affected and do not deteriorate.

For this reason, a system is required to measure the essential variables and data to preserve these conditions and perform the actions needed. Consequently, the Library Mechatronic System is able to measure all this essential information and display it for its proper use. Hence, the main purposes of the system are;

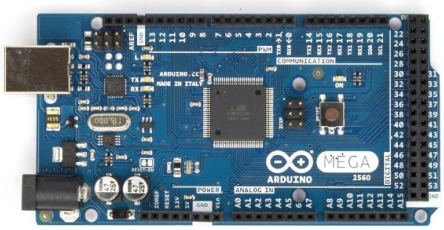
- Detect and count the people that go into and out of the library, making use of proximity sensors.
- Measure the humidity of the space to ensure that the books are kept in the ideal conditions.
- Measure the temperature of the space in order to maintain a proper climate for the users and books.
- Display all this information, including the time and number of people inside the library.

2 Components of the system

In this section we will describe all the system components as well as their respective costs and suppliers.

The components that make up the system are the following:

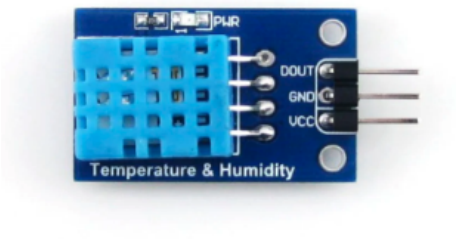
2.1 Arduino Mega 2560



We use the **Arduino Mega**, a microcontroller board based on the ATMega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs) and 16 analog inputs.

2.2 DHT11 Temperature and Humidity Sensor

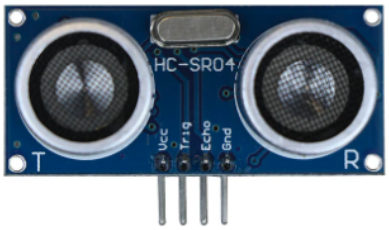
For measuring the temperature and humidity of the library we use the **DHT11 Temperature and Humidity** sensor. It is a composite sensor which contains a calibrated digital signal output of the temperature and humidity.



3 connections to the sensor are only needed. The connections are; Voltage, Ground and Signal.

2.3 HC-SR04 Ultrasonic Sensor Module

For detecting the people that go into an out, the **HC-SR04 Ultrasonic** module is used. It provides 2cm-400cm non-contact measurement function and the ranging accuracy can reach to 3 mm.

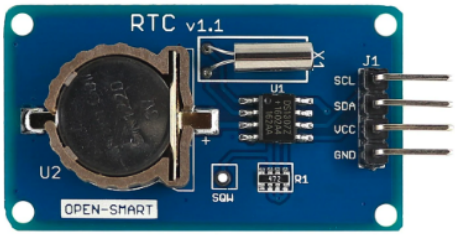


The module includes ultrasonic transmitters, receiver and control circuit. It uses an IO trigger for at least 10us high level signal, then, it sends eight 40 kHz and detect whether there is a pulse signal back. Finally, if the signal is back, through high level, time of high output IO duration is the time from sending ultrasonic tore tuning.

2.4 DS1307 Real Time Clock Module

In order to know the exact time in every moment, the system uses a **DS1307 Real Time Clock** module.

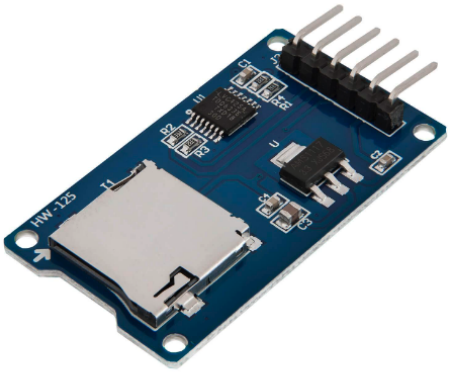
It is a low-power chip. The module provides seconds, minutes, hours, day, date, moth, and year information.



It uses 4 connections; the SCL pin, the SDA, the VCC and GND pins. The 32K and SQW pins are ignored. By oppening the serial monitor, you can see that the module can read the time.

2.5 Micro SD Adapter

For storing all the data, a **Micro SD** adapter is used. Its main function is to transfer data to and from a standard SD card.

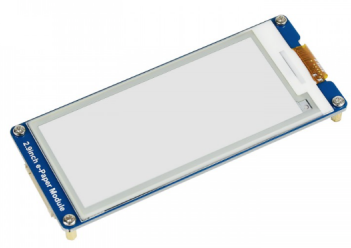


This module uses the standard SPI interface communication, involving SPI buses, MISO, MOSI, SCK, and CS signal pin.

The VCC is connected with the 5V in teh Arduino. Then, the GND to the ground of the Arduino. The CS to pin 14, the SCK to pin 13, MOSI to the pin 11 and MISO to pin 12.

2.6 E-Ink Display Module

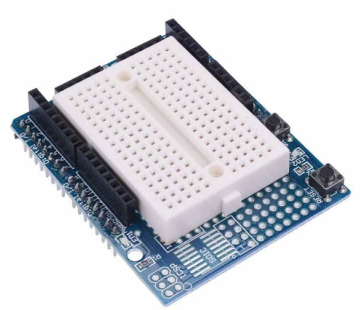
In order to display all the relevant information, the system has a **E-Ink Display** module. It is designed to replicate the appearance of ink on paper.



E-Ink screens are able to hold text and images indefinitely, even without power. It has an ultra low power consumption and power is only required for refreshing.

2.7 Arduino Shield

In order to expand the functionalities of the controller, we use an **Arduino Shield**. It enables us to solder electronics directly on it.



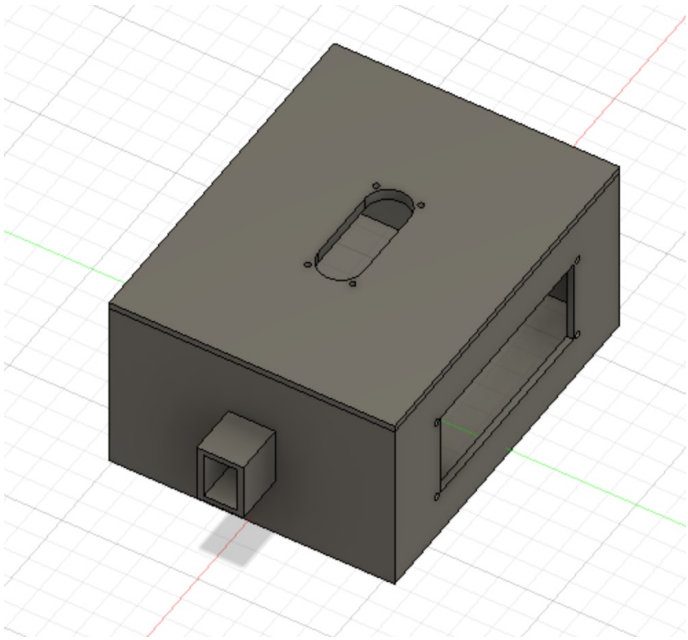
2.8 Power Cord

The system is powered via a power cord.

2.9 Plastic Box

The whole mechatronic system is kept in a 3D designed and printed plastic box that protects all the components and gives the system an aesthetic design.

We designed it using AutoCAD and then printed it using a 3D printer.



We used 113 g of PLA (37 m). The total costs of building the box;

COMPONENT	COST
113 g of PLA	$ 2.20
Electricity & others	$ 0.50
TOTAL	**$2.70**

2.10 Costs and Suppliers

COMPONENT	SUPPLIER	COST
Arduino Mega	Amazon	$ 20.00
DHT11 Temperature and Humidity Sensor	Aliexpress	$ 0.74
HC-SR04 Ultrasonic Sensor	Sparkfun	$ 4.50
DS1307 Real Time Clock Module	Aliexpress	$ 2.30
Micro SD Adapter	Aliexpress	$ 0.63
E-Ink Display Module	Waveshare Vendor	$ 18.99
Arduino Shield	Switch Electronics	$ 3.14
Power Cord	Amazon	$ 15.21
Plastic Box	3D Printed	$ 2.70
TOTAL		**$ 68.21**

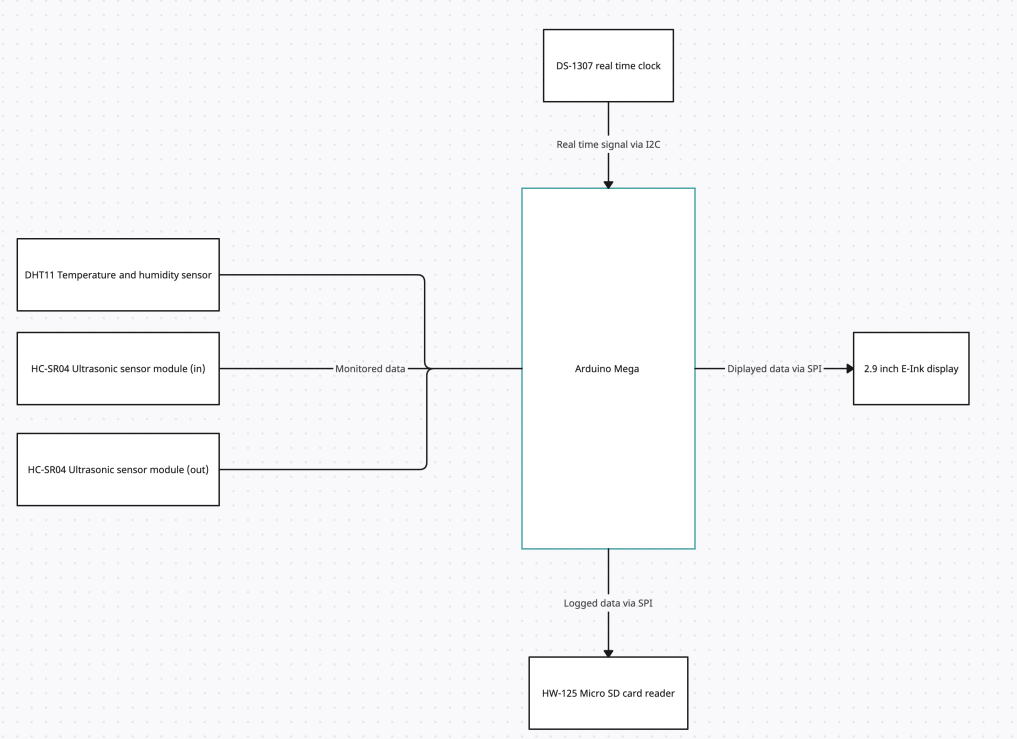
The total cost of the mechatroinc system is approximately $68.00.

It is important to know that some of the components were included directly with the Elegoo Kit. In the next table, you can see which of them were bought externally.

COMPONENT	Elegoo Kit	Bought Externally
Arduino Mega		**X**
DHT11 Temperature and Humidity Sensor	**X**	
HC-SR04 Ultrasonic Sensor	**X**	
DS1307 Real Time Clock Module	**X**	
Micro SD Adapter		**X**
E-Ink Display Module		**X**
Arduino Shield	**X**	
Power Cord	**X**	
Plastic Box		**X**

If we only account for the components that were bought externally, the add-in cost is approximately $42.00.

3 System operation

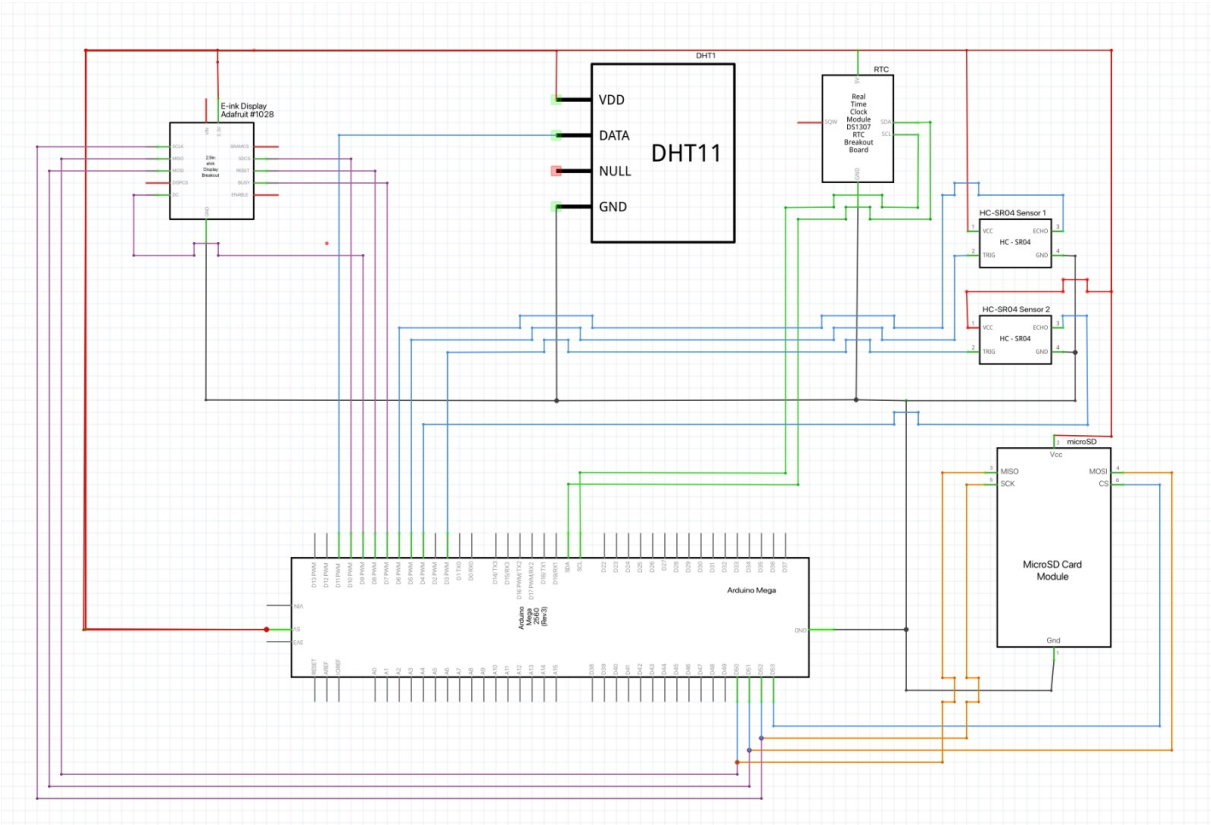


Block diagram

The monitoring device is controlled by an Arduino Mega 2560. When the program is initiated, the display is updated and the real-time clock is updated with the time sent by the computer. Also, during the setup, the SD card reader is initialized to make sure the connection is good. Then the main loop begins. Every iteration, the arduino updates its monitoring variables by calling the different sensors to keep track of the number of people inside the library, the temperature and the humidity level and updates the E-Ink display with the accurate measurements. At the end of the loop, the Arduino checks if it's been five minutes since the last update of the data on the SD card. If it's the case, the data is pushed on the SD card.

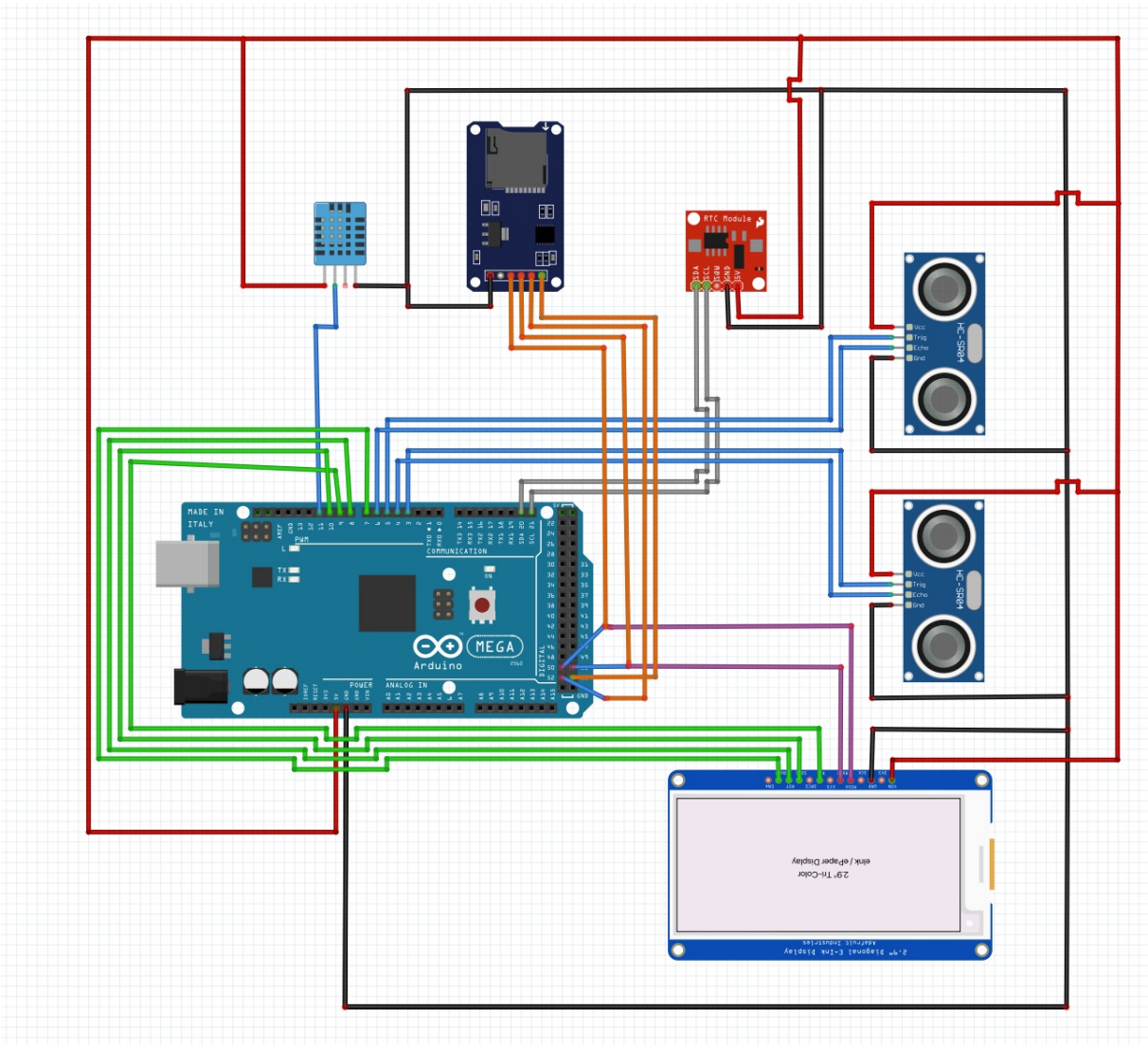
4 Circuit diagram

The Circuit Diagram of the the Library Mechatronic System is the next one:

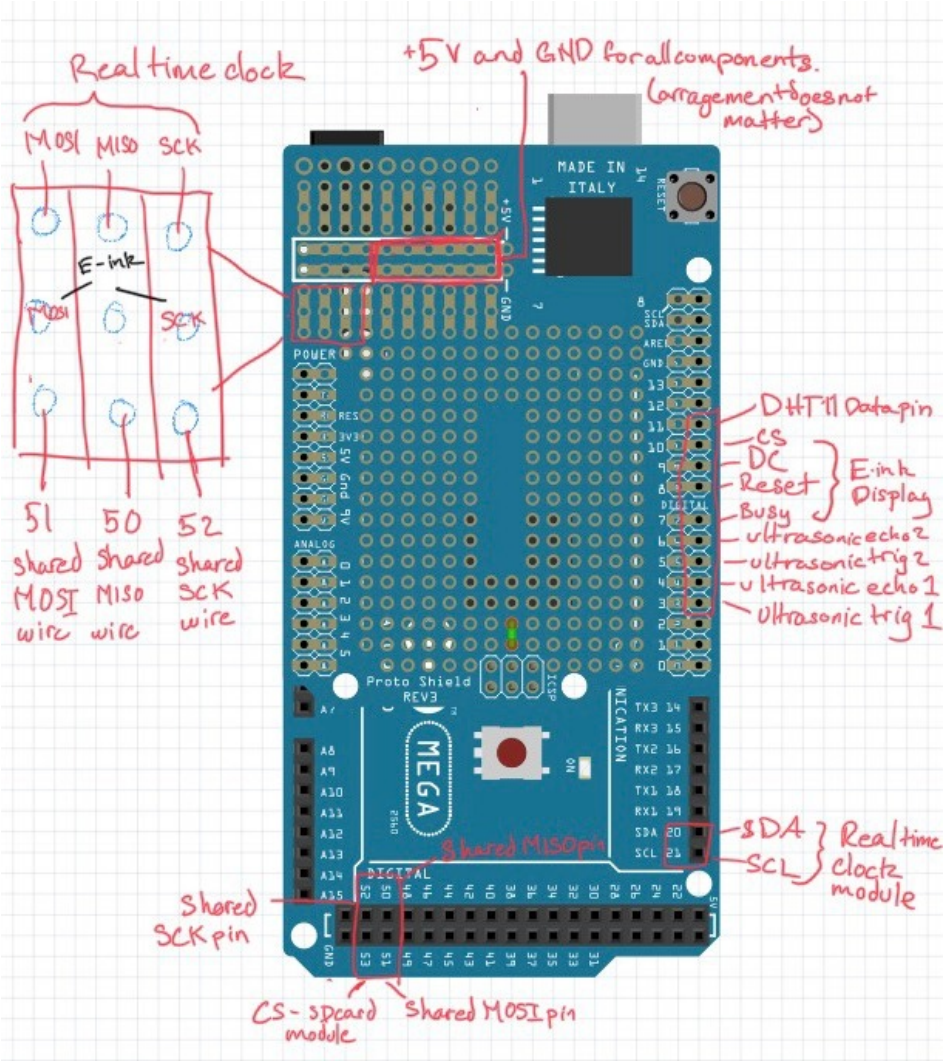


5 Wiring diagram

The wiring diagram showing all the components of the system is the next one:



And the protoshield layout;



6 Instructions for use of the system

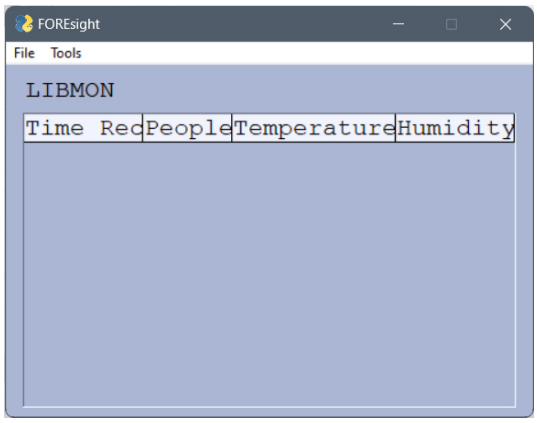
User should first check that the system is wired properly and that no wires are disconnected from their assigned pin.

The system can then be powered up by connecting it to a laptop through USB port or to an outlet with an appropriate power cable. The source code can be found on this GitHub repository.

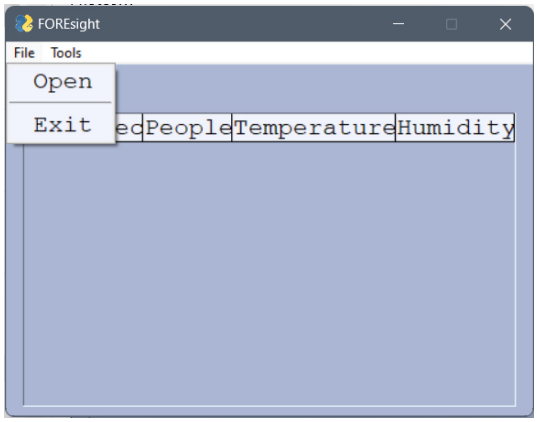
On Github, press the green “Code” button, and then select the “Local” option and “Download ZIP”. VS Code is the recommended software for running the code as it also allows the user to upload to the Arduino using Platform IO as an add-on. User will also need to connect the system to a computer via USB for initial setup.
Once everything is installed and connected, the user can press on the add-on button on the bottom left of VS Code to use Platform IO services. On the home tab, press “Open Project” and open the Prototype folder form the project ZIP file. User then need to modify the platform.ini file in the opened project by adding the full directory paths of the required Arduino libraries on their computer. The user can then access the main.cpp file found in the “src” folder.
After opening main.cpp, the user has access to a taskbar at the bottom left of the VS code window provided by Platform IO. For complete setup, user should first press the “Build” button. If no errors arise and the process is successful, user can press the “Upload” option to upload the code on the Arduino board. The serial monitor can then be used to confirm that the data is effectively written to the SD card and that the distance sensors are working properly. After initial configuration, the system can be powered by an outlet and used in any space the user wish to monitor.

6.1 Instructions for use of FOREsight App

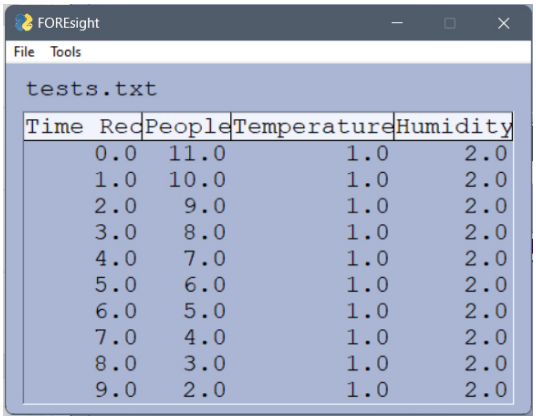
The app FOREsight has a couple of useful applications. Once the app is open and running, the user will notice a menu in the top bar. This menu is made up of “File” and “Tools”.



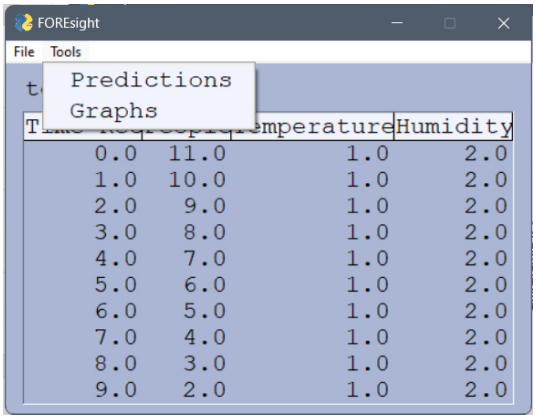
Under “File”, there are 2 options to choose from, “Open” and “Exit”.



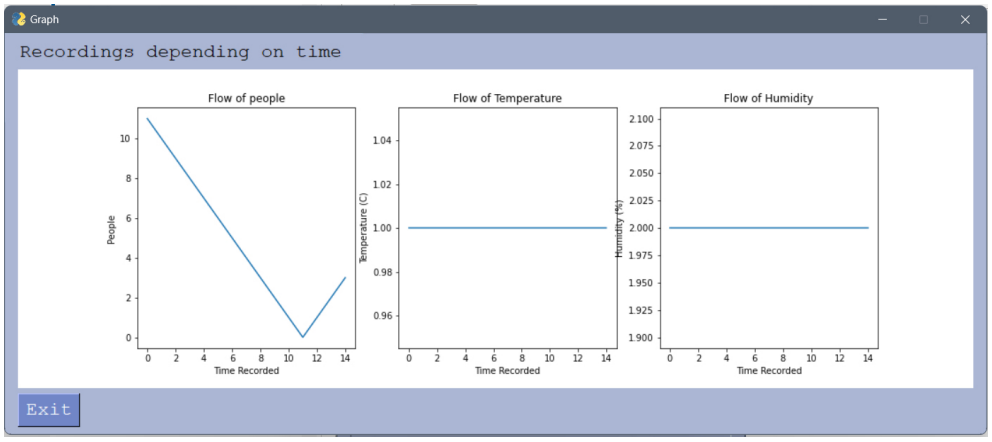
The “Open” action will then prompt the user to choose the file containing the recordings produced by the Library Monitor. The “Exit” option will close the app. This can also be achieved by pressing the “X” button in the top right corner. Once opened, the file will be shown on the app’s primary interface, filling the table’s columns with the appropriate recordings: “Time Rec”, “People”, “Temperature”, and “Humidity”.



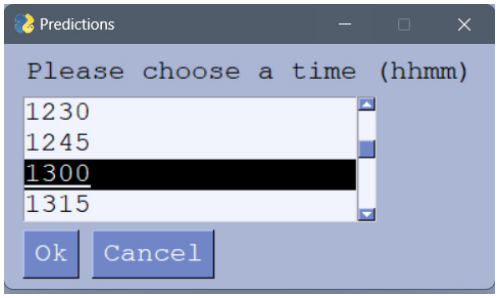
Once the file has been opened this way, the user can select the “Tools” of his choosing.

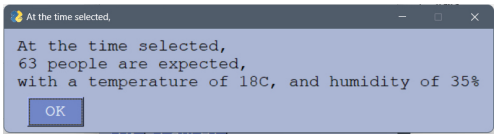


Under “Tools”, the user can find “Predictions” and “Graphs”. The “Predictions” action will prompt a pop-up asking the user to select a time they plan to get to the library. Based on this user input, the app will produce another pop-up with the prediction of the number of people (how busy the library will be), the temperature, and the humidity at that selected time.



These predictions are based on the extrapolation of data from the recordings obtained by the system. Thus, the user can find the optimal time and conditions for their trip to the library. The “Graphs” action will produce a pop-up of three subplots, each detailing how the three recordings, number of people, temperature, and humidity, change throughout the day based on time.





NB: All of the above graphs and predictions are based on test-case data for verification of the app. None of the data in this section was collected in a real-time environment.

7 Code Main System

The code of the system commented:

```
#include <Arduino.h>
#include <string.h>

// Temperature and humidty sensor
#include "DHT.h"
#define DHTPIN 11
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
float temperature;
float humidity;

// Ultrasonic sensor
#include "SR04.h"
int TRIG_PIN = 5; // assign pin 5 and 6 to first ultrasonic sensor
int ECHO_PIN = 6;
SR04 dist_sensor_in = SR04(ECHO_PIN,TRIG_PIN); // first sensor

int TRIG_PIN_2 = 3; // assign pin 3 and 4 to second ultrasonic sensor
int ECHO_PIN_2 = 4;
SR04 dist_sensor_out = SR04(ECHO_PIN_2,TRIG_PIN_2); // second sensor
// initialize variables for sensors
int distance_in; // distance measured by sensor 1
int distance_out; // distance measured by sensor 2
int counter = 0; // number of people inside
bool previous_state_in = false;
bool previous_state_out = false;
const int distance_treshold = 30; // decision threshold

// Display
#include <epd2in9_V2.h>
#include <epdpaint.h>
#include <imagedata.h>

#define COLORED     0
#define UNCOLORED   1

unsigned char image[1024];
Paint paint(image, 0, 0);    // width should be the multiple of 8
Epd epd;
unsigned long current_time;
unsigned long time_now_s;

//Real time clock
#include <Wire.h>
#include <DS3231.h>
DS3231 clock;
RTCDateTime dt;
String time;
String date;
String filename = String("data.txt");

// SD card
#include <SPI.h>
#include <SD.h>
```

```
55
56
57  File myFile;
58
59  void initialization_sd(){
60    // Create a file on the SD card
61    while (!Serial) {
62      ; // wait for serial port to connect. Needed for native USB port only
63    }
64    Serial.print("Initializing SD card...");
65
66    if (!SD.begin(53)) {
67      Serial.println("initialization failed!");
68      // Verify the SD card is present and can be initialized.
69      while (1);
70    }
71    Serial.println("initialization done.");
72
73    if (SD.exists("data.txt")) { // if the file exists, append data to it:
74      Serial.println("data.txt exists. Appending data.");
75    } else {
76      Serial.println("data.txt doesn't exist. Creating a new file.");
77      myFile = SD.open("data.txt", FILE_WRITE); // else create a file if none detected
78      myFile.println("Date, Time, Temperature, Humidity, Number of people");
79      myFile.close();
80    }
81  }
82  // function to write data to SD card
83  void write_data( String date, String time, float temperature, float humidity, int counter){
84    myFile = SD.open("data.txt", FILE_WRITE);
85    if (myFile) {
86      myFile.println(date + String(",") + time + String(",") + String(temperature) + String(",") + String(humidity) + String(",") + String(counter));
87      myFile.close();
88    } else {
89      Serial.println("error opening data.txt");
90    }
91  }
92
93  // function to initialize e-paper screen
94  void init_screen()
95  {Serial.begin(9600);
96    if (epd.Init() != 0) {
97        Serial.print("e-Paper init failed");
98        return;
99    }
100
101   epd.ClearFrameMemory(0xFF);   // bit set = white, bit reset = black
102   epd.DisplayFrame();
103
104
105   if (epd.Init() != 0) {
106       Serial.print("e-Paper init failed ");
107       return;}
108   epd.SetFrameMemory_Base(IMAGE_DATA);
109   epd.DisplayFrame();
110
111   current_time = millis();
112 }
113
114
```

```
// function to update the screen with new measurements every 5 min
void loop_screen(String time, String counter, String temperature, String humidity) {
  // Update for time
  char time_string[] = {'C','u','r','r','e','n','t',' ','t','i','m','e',':', '0', '0', ':', '0', '0','\0'};
  time_string[13] = time[0];
  time_string[14] = time[1];
  time_string[16] = time[3];
  time_string[17] = time[4];
  paint.SetWidth(32);
  paint.SetHeight(238);
  paint.SetRotate(ROTATE_90);

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(0, 4, time_string, &Font16, COLORED);
  epd.SetFrameMemory_Partial(paint.GetImage(), 100, 00, paint.GetWidth(), paint.GetHeight());
  // update for temperature
  char temp_string[] = {'T', 'e', 'm', 'p', 'e', 'r', 'a', 't', 'u', 'r', 'e', ':', ' ', '-', '-', '.', '-', 'F', '\0'};

  temp_string[13] = temperature[0];
  temp_string[14] = temperature[1];
  temp_string[16] = temperature[3];

  paint.SetWidth(32);
  paint.SetHeight(220);
  paint.SetRotate(ROTATE_90);

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(0, 4, temp_string, &Font16, COLORED);
  epd.SetFrameMemory_Partial(paint.GetImage(), 70, 0, paint.GetWidth(), paint.GetHeight());
  // update for humidity
  char humidity_string[] = {'H', 'u', 'm', 'i', 'd', 'i', 't', 'y', ':', ' ', '-', '-', '.','-','-', '%', '\0'};
  humidity_string[10] = humidity[0];
  humidity_string[11] = humidity[1];
  humidity_string[13] = humidity[3];
  humidity_string[14] = humidity[4];



  paint.SetWidth(32);
  paint.SetHeight(220);
  paint.SetRotate(ROTATE_90);

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(0, 4, humidity_string, &Font16, COLORED);
  epd.SetFrameMemory_Partial(paint.GetImage(), 40, 0, paint.GetWidth(), paint.GetHeight());
  // update for person counter
  char counter_string[] = {'P', 'e', 'o', 'p', 'l', 'e', ' ', 'i', 'n', 's', 'i', 'd', 'e', ':', ' ', '0', '0', '0', '0', '\0'};

  counter_string[15] = counter[0];
  counter_string[16] = counter[1];
  counter_string[17] = counter[2];
  counter_string[18] = counter[3];

  paint.SetWidth(32);
  paint.SetHeight(240);
  paint.SetRotate(ROTATE_90);

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(0, 4, counter_string, &Font16, COLORED);
  epd.SetFrameMemory_Partial(paint.GetImage(), 10, 0, paint.GetWidth(), paint.GetHeight());
  epd.DisplayFrame_Partial();
}

// function of the sensors that update the counter when persons go in and out of detected area
void call_sensor(){
    distance_in = dist_sensor_in.Distance(); // Detect when a person enters
  if (distance_in < distance_treshold){
  if (!previous_state_in){
    counter++; // only adds 1 if person just entered the field
    previous_state_in = true;
    }
  }
  else{
    previous_state_in = false;
  }

  Serial.print("Distance sensor in: "); // for debugging and knowing the rate of detection of sensor
  Serial.print(distance_in);
  Serial.print("\n");
```

```
194
195   distance_out = dist_sensor_out.Distance(); // Detect when a person leaves
196   if (distance_out < distance_treshold){
197     if (!previous_state_out){
198       counter--; // only substract 1 if person just entered the field
199       previous_state_out = true;
200     }
201   }
202   else{
203     previous_state_out = false;
204   }
205
206   Serial.print("Distance sensor out: "); // for debugging and knowing the rate of detection of sensor
207   Serial.print(distance_out);
208   Serial.print("\n");
209 }
210 // setup code goes here
211 void setup() {
212   Serial.begin(9600);
213   Serial.println("Starting...");
214   clock.begin(); // initialize clock
215   dht.begin(); // initialize temp and humidity sensor
216   clock.setDateTime(__DATE__, __TIME__);    // Set the RTC to the compile time
217   Serial.print("Time: ");
218   Serial.println(__TIME__);
219   initialization_sd(); // initialize SD card reader
220   delay(200);
221   init_screen(); // initialize screen
222   Serial.print("Setup completed");
223 }
224
225 // loop code goes here
226 void loop() {
227
228   call_sensor(); // ultrasound sensors measurements
229
230   dt = clock.getDateTime(); // get data from clock module
231   if ( dt.second % 315 == 0 ) // every 5 min
232   {
233     Serial.println("Reading sensor");
234     // Get time
235     time = String(dt.hour/10) + String(dt.hour % 10) + String(":") + String(dt.minute / 10) + String(dt.minute % 10 )+ String(":") + String(dt.second / 10) +
236     date = String(dt.day) + String("/") + String(dt.month);
237
238     // Get temperature and humidity
239     humidity = dht.readHumidity(true);
240     //read temperature in Fahrenheit
241     temperature = dht.readTemperature(true);
242
243     // Write to SD card
244     Serial.println("Writing to SD card");
245     write_data(date, time, temperature, humidity, counter);
246     Serial.println("Data written to file:");
247     Serial.println(date + String(",") + time + String(",") + String(temperature) + String(",") + String(humidity) + String(",") + String(counter));
248   }
249   // string for person counter
250   String counter_string = String(counter/1000) + String(counter/100 % 10) + String(counter/10 % 10) + String(counter % 10);
251     loop_screen(String(time), String(counter), String(temperature), String(humidity)); // update screen with current values
252   delay(1000); // idk if we need this
253 }
```

8 Code FOREsight App

The code for the FOREsight App is the next one:

```
import PySimpleGUI as sg
from pathlib import Path
import pandas as pd

import os
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

import warnings
warnings.filterwarnings('ignore')
```

```
menu_layout = [
    ['File', ['Open', '---', 'Exit']],
    ['Tools', ['Predictions', 'Graphs']]
]
data = []
layout = [
    [sg.Menu(menu_layout)],
    [sg.Text('LIBMON', key = '-DOCNAME-')],
    [sg.Table(headings = ["Time Rec", "Temperature", "Humidity", "People"],
                values = data,
                expand_x = True,
                hide_vertical_scroll = True,
                key = '-TABLE-')]]
# Create the Window
window = sg.Window('FORESight', layout)

# Event Loop to process "events" and get the "values" of the inputs
while True:
    event, values = window.read()
    if event == sg.WIN_CLOSED or event == 'Exit': # if user closes window or clicks cancel
        break

    if event == 'Open':
        file_path = sg.popup_get_file('Insert LIBMON file', no_window=True, default_extension='txt')
        if file_path:
            file = Path(file_path)
            data = open(file, "r")
            window['-DOCNAME-'].update(file_path.split('/')[-1])
            i=0
            time_array = []
            temp_array = []
            hum_array = []
            people_array = []
            for x in data:
                line = x
                time = int(line[5:7])*100+int(line[8:10])
                time_array.append(time)
                temp = float(line[14:19])
                temp_array.append(temp)
                hum = float(line[20:25])
                hum_array.append(hum)
                people = int(line[26:])
                people_array.append(people)
                update.append([int(line[5:7])*100+int(line[8:10]), float(line[14:19]), float(line[20:25]), int(line[26:])])
                i = i+1
            window['-TABLE-'].update(update)

    if event == 'Graphs':
        graph_pop(window, time_array, temp_array, hum_array, people_array)

    if event == 'Predictions':
        pred_pop(window, time_array, temp_array, hum_array, people_array)

window.close()
```

9 Captured data

For capturing the data, we placed the Mechatronic System in the Franklin Wilkins Building Library.

However, the power cord was a trip risk and the sensor had to be moved from its position at the swipe gates. Because it was moved the sensors could no longer record people entering and exiting, only the few that passed by it on one side. This has affected our data collection, and prevented us from using the predictive function in the FOREsight app.

Figure 1: System in center of swipe gates

Figure 2: New position

In **Figure 1** you can see the correct position, in which data was registered for both sides; registering people entering and exiting. In **Figure 2**, you can see the new position, where the sensor was moved and could only register people entering one side.

9.1 Graph Data

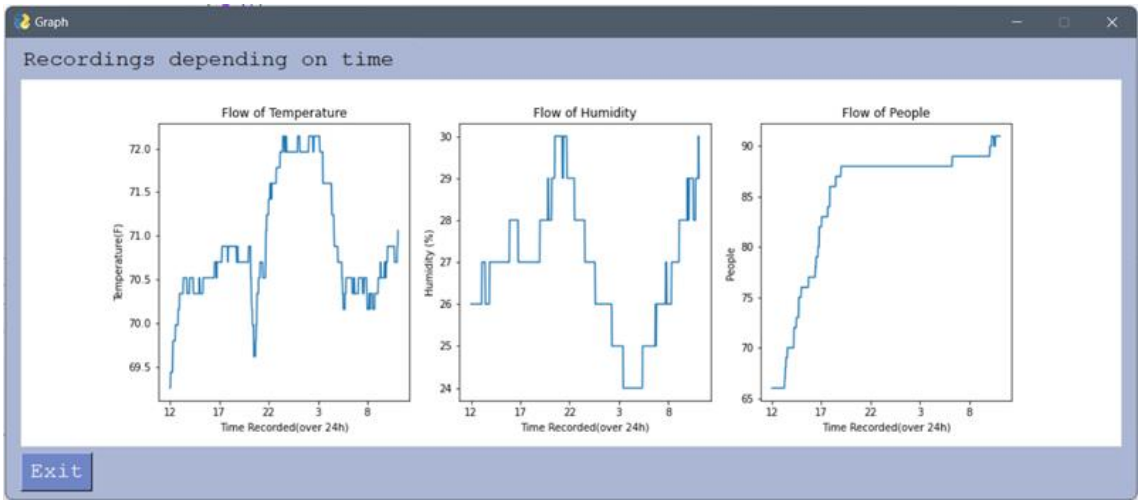


Figure 3: Graphed Humidity, Temperature, and People passing for 24 hours starting from 12:00pm.

The graphed data for the number of people is not accurate to the number of people who entered and exited the FWB library over 24 hours. As stated in the previous section on data collection, the library staff did not allow the sensor to sit between swipe gates as the extension cord to the power source was a trip hazard. The Mechatronic system was set off to the side and unable to properly count people passing through the gates.

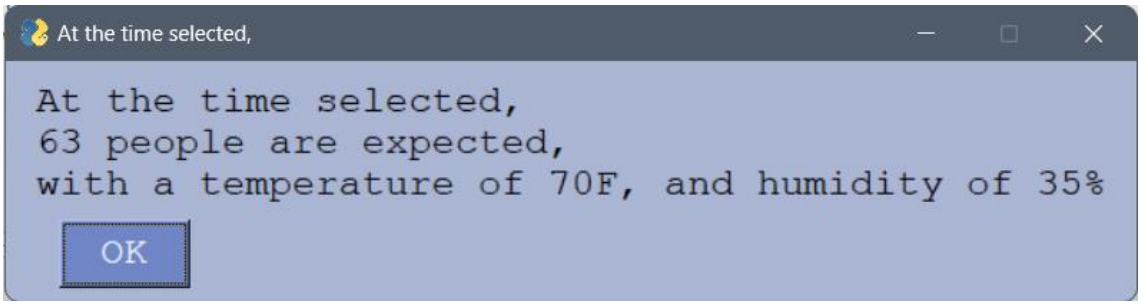


Figure 4: Prediction of values based on time selected

In **Figure 4** the predictions are based on an averaging of all values for the selected time. Of course, the number of people predicted is not accurate.

10 References

The external sources and references we have used are;

- Arduino Store
- E-Ink Display Vendor
- Elegoo, The most important starter kit manual.