**MODULATED OCCUPANCY DETECTION SYSTEM**

By

|  |  |
| --- | --- |
| RAHUL ANIL NAIR | 19BEC1431 |

A project report submitted to

**Prof. JOHN SAHAYA RANI ALEX**

**SCHOOL OF ELECTRONICS ENGINEERING**

In partial fulfillment of the requirements for the course of

**ECE3003 – MICROCONTROLLER AND ITS APPLICATIONS**

in

**B. TECH ELECTRONICS AND COMMUNICATION ENGINEERING**



**VANDALUR – KELAMBAKKAM ROAD**

**CHENNAI – 600127**

**JUNE 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report entitled “**OCCUPANCY DETECTION SYSTEM”** is a bonafide work of  **RAHUL ANIL NAIR - 19BEC1431,** who carried out the Project work under my supervision and guidance for **ECE3003 - MICROCONTROLLER AND ITS APPLICATIONS**

**Prof JOHN SAHAYA RANI ALEX**

Associate Professor

School of Electronics Engineering(SENSE),

VIT University, Chennai

Chennai – 600 127.

**ABSTRACT**

Elevator Safety is a major concern for builders and architects. A lift can carry only a certain limit of people, and when more people enter exceeding the threshold limit, the lift gets overloaded. This causes damage to the lift mechanism and can also be viewed as a potential threat in the long run if the elevator is left unserviced.

Our Arduino Uno R3 based Occupancy Detection System’s main objective is to detect people coming in and going out of the building using motion sensors and measure the shift in weight change.

The occupancy sensors are indoor motion detecting devices which are used to detect presence of a person. Based on occupancy in a room or building, innumerable decisions can be made such as controlling lights, calculating headcount, automatic door control and AC control.

During a stop at a floor, the elevator car doors will open for a short period of time to allow passengers to enter or exit the elevator car. The doors will then close again and during this period, people enter and exit the lift, i.e. the total count of people inside the lift changes. The load cell obtains the shift in the weight which the Arduino then processes and displays the weight on the lcd display.

Hence, we can easily keep track of occupancy and keep the lift safe from any hazard of overload by stopping people from entering further.

Thus, with facilitating the people on the lift occupancy count, it also serves as a means for ensuring safety.

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof John Sahaya Rani Alex**, Associate Professor, School of Electronics Engineering, for her consistent encouragement and valuable guidance pleasantly offered to us throughout the course of the project work.

We are extremely grateful to **Dr. Sivasubramanian A. -** the Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department - **Dr. Vetrivelan P.** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity that they provided us in undergoing this course in such a prestigious institution.

**RAHUL ANIL NAIR**

**Table of Contents**

1. **Introduction** 
   1. Objectives **6**
   2. Features  **6**
   3. Scope **6**
2. **Occupancy Detection System using Arduino Uno R3** 
   1. Block diagram **7**
   2. Component description **8**
   3. Original prototype **11**
   4. Schematic diagram **-** Tinkercad **12**
   5. Software specifications **12**
   6. Feasibility

2.6.1 Economic Feasibility 13

2.6.2 Technical Feasibility 13

1. **System implementation and analysis**

**3.1 System Implementation 14**

3.1.1 Program Code (Arduino) **14**

3.1.2 Output Screenshot **19**

**3.2 Program Code (Tinkercad) 20**

**3.3 Code analysis 20**

**3.4 Hardware Connections 21**

**4 Summary and Conclusion 22**

**5 Future Enhancements 23**

**6 References 24**

**7 Bio-Data 25**

**INTRODUCTION**

**1.1 OBJECTIVES**

* Designing the OCCUPANCY DETECTION SYSTEM using Arduino UNO R3
* Measuring the total weight of the people present inside an elevator.
* Controlling the number of people entering a lift with the buzzer indication.
* Establishing IR sensor connection with Arduino Uno R3 in order to sense the number of people entering.
* Establishing 16x2 LCD connection with Arduino Uno R3 for the same.

**1.2 FEATURES**

* Use of Hx711 ADC module
* Sensory devices to sense various things
* Use of external devices such as load cell, buzzer, servo motor, LCD
* IR communication module to transmit and receive data

The efficient coding is done on a software tool called the Arduino IDE software, and another software that is being used is the Tinkercad software for simulating the components we could not order before hand.

**1.3 SCOPE**

* Can be used efficiently to alert if there is elevator overloading
* Can be used to count the visitors of an auditorium, hall, offices, mall, sports etc.
* Can be used as an integral part of the security system in high confidential areas.
* Can be used in Parking lots.
* High precision and accuracy can be achieved through it.
* Since it is quite compact it can be easily implemented anywhere.

**OCCUPANCY DETECTION SYSTEM USING ARDUINO UNO R3**

**2.1 BLOCK DIAGRAM**

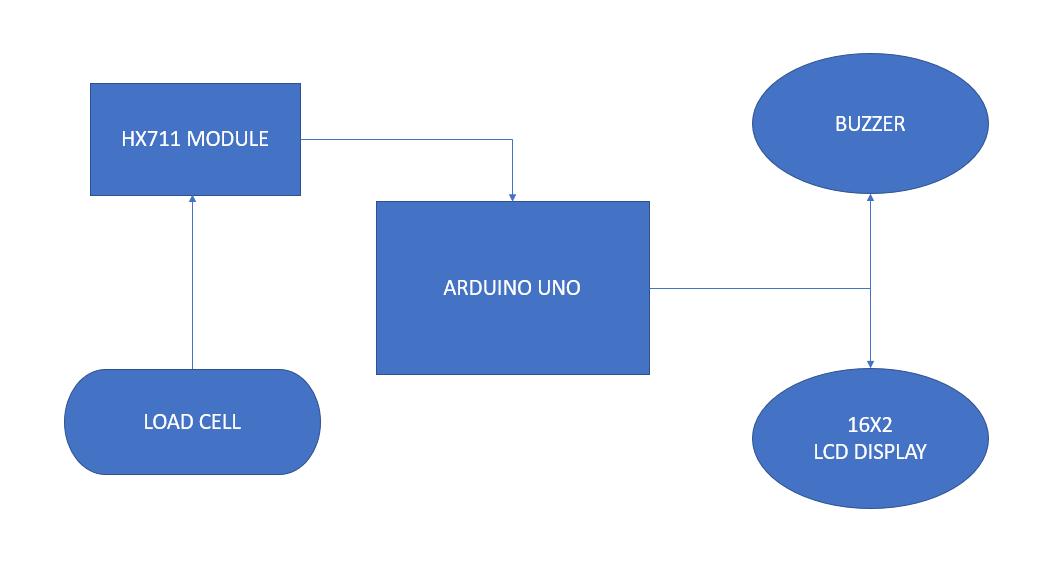
The four main features of the basic block diagram (given below) are:

* ARDUINO UNO R3
* 16x2 LCD DISPLAY
* LOAD CELL
* HX711 AMPLIFIER MODULE

EXTRA FEATURES:

* BUZZER
* The IR (INFRARED) sensor connected to Arduino Uno R3
* LED
* MICRO SERVO

***Schematic Block Diagram***

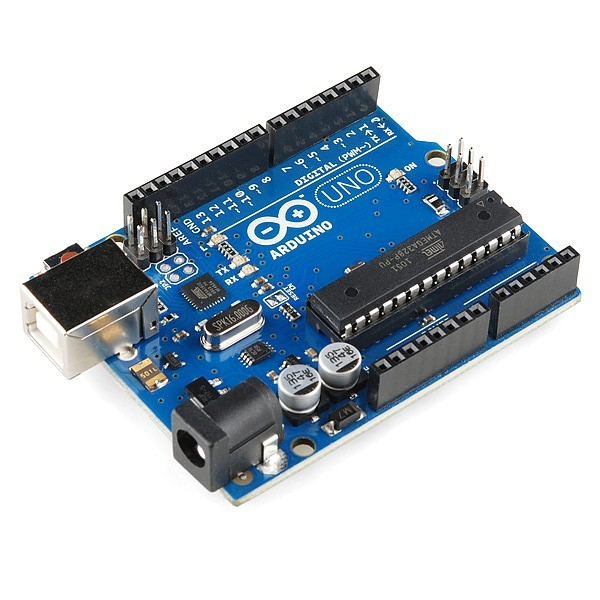
****

**2.2COMPONENT DESCRIPTION**

**1) ARDUINO UNO R3**

The Arduino Uno is an open source microcontrollerboard based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards(shields) and other circuits.

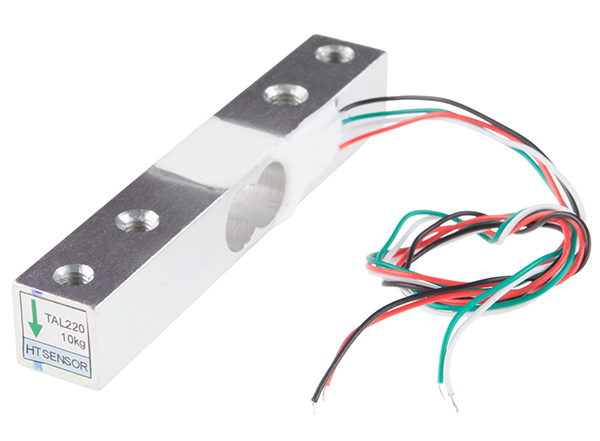
The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USBcable. It can be powered by the USB cable or by an external 9-V battery, though it accepts voltages between 7 and 20 volts.



**2) LOAD CELL:**

A load cell is a type of transducer, specifically a force transducer. It converts a force such as tension, compression, pressure, or torque into an electrical signal that can be measured and standardized.

A load cell works by converting mechanical force into digital values that the user can read and record. The inner working of a load cell differs based on the load cell that you choose. There are hydraulic load cells, pneumatic load cells, and strain gauge load cells. Strain gauge load sensors are the most commonly used among the three. Strain gauge load cells contain strain gauges within them that send up voltage irregularities when under load. The degree of voltage change is covered to digital reading as weight.



**3) 16X2 LCD DISPLAY**

LCD modules are very commonly used in most embedded projects, the reason being its cheap price, availability and programmer friendly. Most of us would have come across these displays in our day to day life, either at PCO’s or calculators. The appearance and the pinouts have already been visualized above now let us get a bit technical.

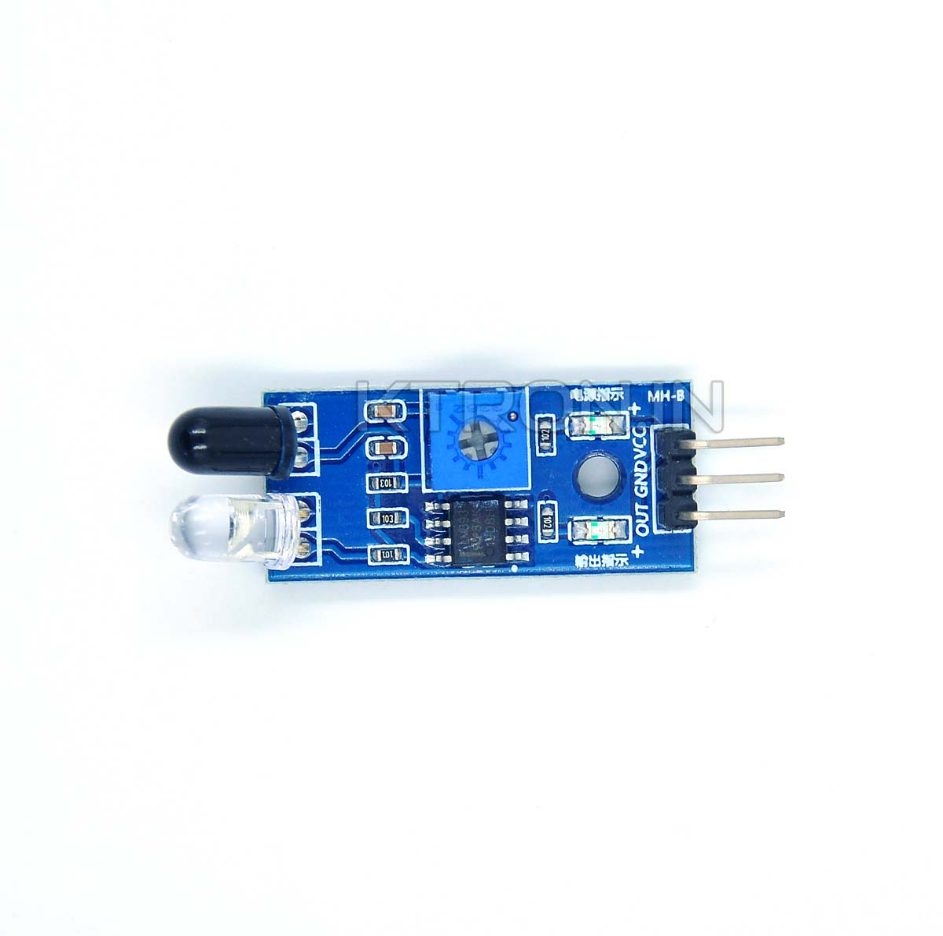
16×2 LCD is named so because; it has 16 Columns and 2 Rows. There are a lot of combinations available like, 8×1, 8×2, 10×2, 16×1, etc. but the most used one is the 16×2 LCD. So, it will have (16×2=32) 32 characters in total and each character will be made of 5×8 Pixel Dots.



**4) IR SENSOR:**

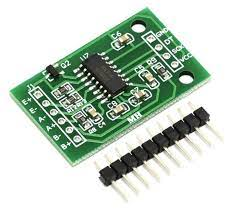
An infrared (IR) sensor is an electronic device that measures and detects infrared radiation in its surrounding environment. While measuring the temperature of each color of light (separated by a prism), he noticed that the temperature just beyond the red light was highest. IR is invisible to the human eye, as its wavelength is longer than that of visible light (though it is still on the same electromagnetic spectrum). Anything that emits heat (everything that has a temperature [above around five degrees Kelvin](https://www.livescience.com/50260-infrared-radiation.html)) gives off infrared radiation.

There are two types of infrared sensors: active and passive. Active infrared sensors both emit and detect infrared radiation. Active IR sensors have two parts: a light emitting diode (LED) and a receiver. When an object comes close to the sensor, the infrared light from the LED reflects off of the object and is detected by the receiver. Active IR sensors act as [proximity sensors](https://www.fierceelectronics.com/sensors/what-a-proximity-sensor), and they are commonly used in obstacle detection systems (such as in robots).



**5) ADC HX711**

HX711 module is a Load Cell Amplifier breakout that allows you to easily read load cells to measure weight. It is an electronic scale module, whose working principle is to convert the measured changes in resistance value changes, through the conversion circuit into electrical output. The load cell amplifier is used to get measurable data out from a load cell and strain gauge.



**6) BUZZER:**

A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

****

**2.3 Photographs of the Original Prototype:**

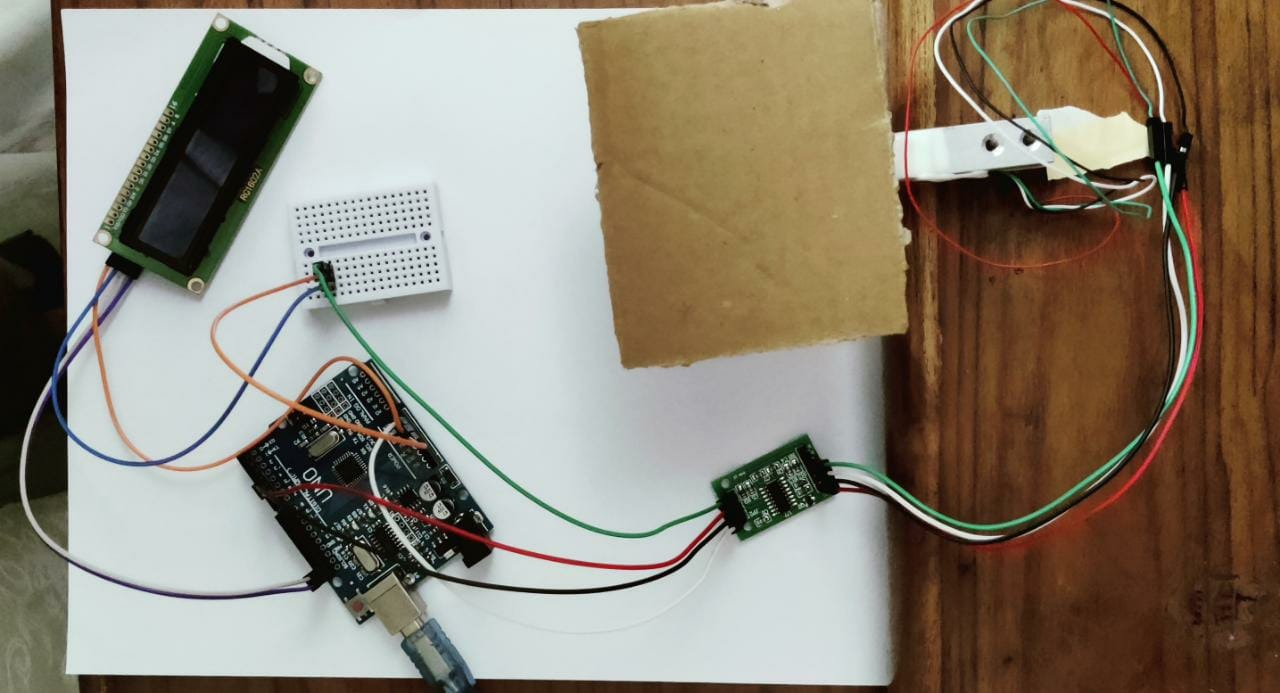
****

Figure 1 Hardware Set-Up

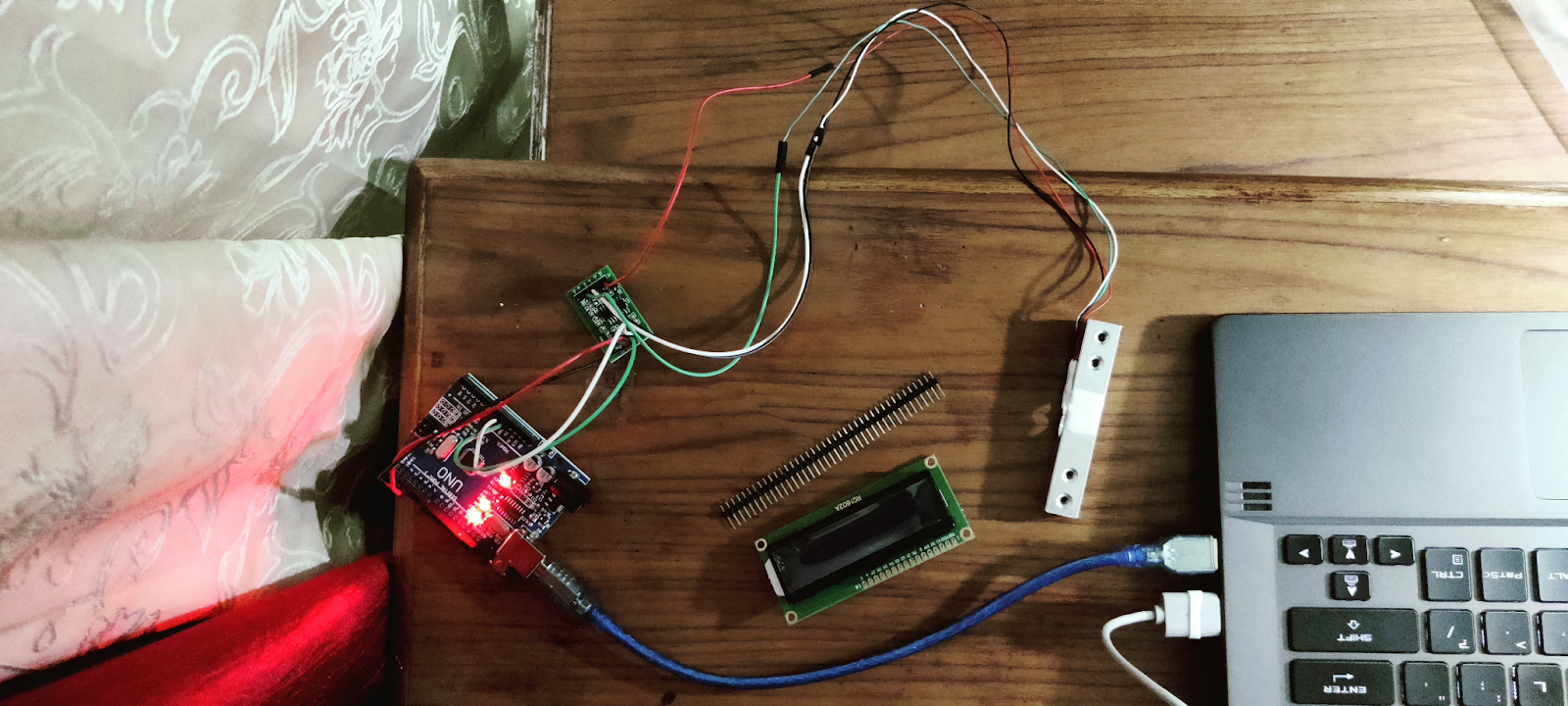
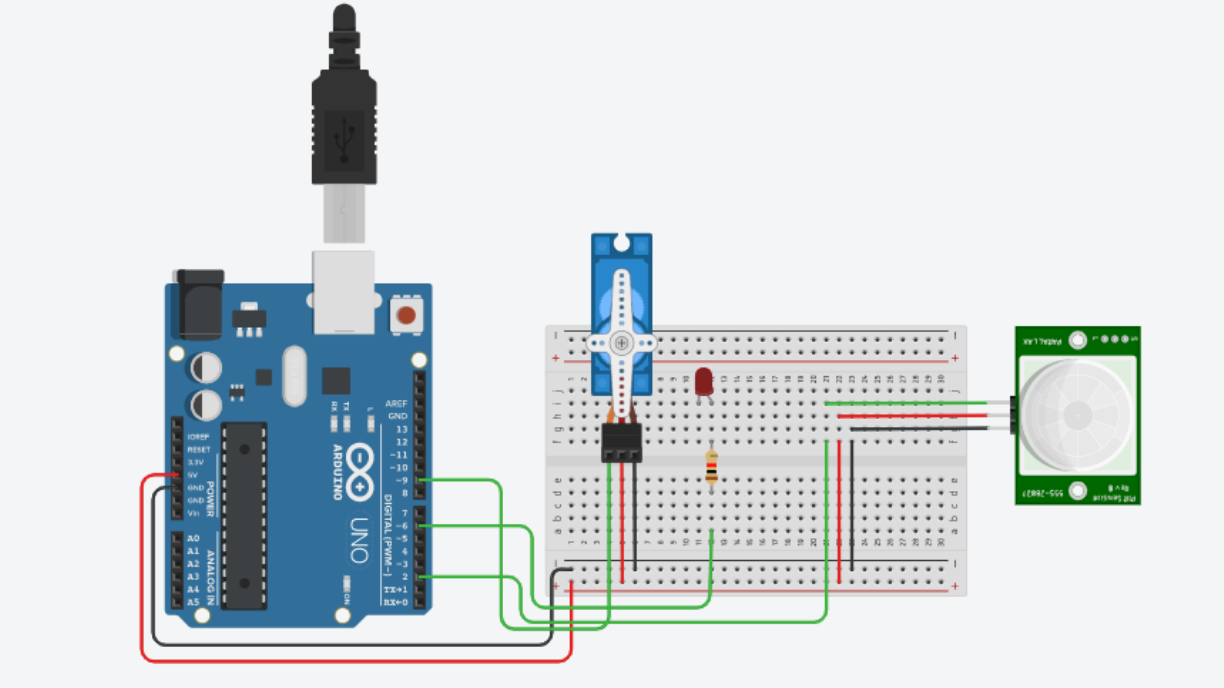
****

Figure 2 Working Hardware

**2.4 SCHEMATIC DIAGRAM (TINKERCAD)**

******

Since we could not purchase the PIR sensor, we decided to build it on Tinkercad.

PIR stands for Passive InfraRed, which describes the technology inside—it passively detects infrared light levels (unlike an infrared camera that may also emit infrared light in order to capture its reflection). The white dome is a lens that expands the IR detector's field of vision. The sensor reports a LOW signal by default, reads the amount of ambient infrared light coming in, and then triggers a HIGH signal for a certain period of time when the light levels change, indicating movement. It can tell you whether or not there is movement in a scene, but cannot detect distance— for that you might consider a type of analog input sensor called an ultrasonic rangefinder.

To optionally build the physical circuit, we would gather up the Arduino Uno board, USB cable, solderless breadboard, an LED, resistor (any value from 100-1K), PIR motion sensor, and breadboard wires.

***TINKERCAD SOFTWARE SIMULATION LINK***

[**https://www.tinkercad.com/things/eAmSvD15ICy-pir-sensor-micro/editel?sharecode=mqBnzJYHmUVOoN8xed7McieFkj0NSEDldQNMlxiIqUA**](https://www.tinkercad.com/things/eAmSvD15ICy-pir-sensor-micro/editel?sharecode=mqBnzJYHmUVOoN8xed7McieFkj0NSEDldQNMlxiIqUA)

**2.5 SOFTWARE SPECIFICATIONS**

* ARDUINO IDE SOFTWARE
* TINKERCAD SOFTWARE

**2.6 FEASIBILITY**

**2.6.1 ECONOMIC FEASIBILITY**

* The Arduino Uno R3 costs around Rs. 750.
* The Load Cell along with HX711 costs about Rs.560.
* Cost for the 16x2 LCD display is Rs. 300.
* Jumper wires took about Rs. 140.
* The Breadboard costed us around Rs. 160.
* The Buzzer would have costed us around Rs. 100. But we were unable to order it due to lockdown.
* Total cost of the project would thus approximately account to Rs. 1910- Rs.2110
* This project is quite cheap and can be implemented anywhere to increase the security and convenience of the visitor as well as the lift.

**2.6.2 TECHNICAL FEASIBILITY**

* The Arduino Uno R3 is an open source microcontroller board based on the ATmega328 chip.
* Hardware being used are easily available and available in the market at low rates.
* Components used are easy to handle and work with.
* Repairing and preparation cost is very low.

**SYSTEM IMPLEMENTATION AND ANALYSIS**

**3.1 SYSTEM IMPLEMENTATION:**

**3.1.1 PROGRAM CODE (Arduino Uno R3)**

#include <HX711\_ADC.h>

#if defined(ESP8266)|| defined(ESP32) || defined(AVR)

#include <EEPROM.h>

#endif

#define THRESHOLD 50

#define BUZZER    13

unsigned long startTime = 0;

bool beep = false;

//pins:

const int HX711\_dout = 4; //mcu > HX711 dout pin

const int HX711\_sck = 5; //mcu > HX711 sck pin

//HX711 constructor:

HX711\_ADC LoadCell(HX711\_dout, HX711\_sck);

const int calVal\_eepromAdress = 0;

unsigned long t = 0;

void setup() {

  Serial.begin(57600);

  delay(10);

  Serial.println();

  Serial.println("Starting...");

  LoadCell.begin();

  unsigned long stabilizingtime = 2000; // preciscion right after power-up can be improved by adding a few seconds of stabilizing time

 boolean \_tare = true; //set this to false if you don't want tare to be performed in the next step

  LoadCell.start(stabilizingtime, \_tare);

  if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {

    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");

    while (1);

  }

  else {

    LoadCell.setCalFactor(1.0); // user set calibration value (float), initial value 1.0 may be used for this sketch

    Serial.println("Startup is complete");

  }

  while (!LoadCell.update());

  calibrate(); //start calibration procedure

  pinMode(BUZZER, OUTPUT); //Setting the buzzer to output

  startTime = millis();

}

void loop() {

 if (LoadCell.update()) {

      float i = LoadCell.getData();

      Serial.print(i);

      if(i > THRESHOLD){

         Serial.print(" Overweight");

         if((unsigned long)(millis() - startTime) > 1000){

          beep = !beep;

          pinMode(BUZZER, beep);

          startTime = millis();

         }

      }

      Serial.println(" ");

    }

// receive command from serial terminal

if (Serial.available() > 0) {

    char inByte = Serial.read();

    if (inByte == 't') LoadCell.tareNoDelay(); //tare

    else if (inByte == 'r') calibrate(); //calibrate

    else if (inByte == 'c') changeSavedCalFactor(); //edit calibration value manually

  }

  // check if last tare operation is complete

  if (LoadCell.getTareStatus() == true) {

    Serial.println("Tare complete");

  }

}

void calibrate() {

  Serial.println("\*");

  Serial.println("Start calibration:");

  Serial.println("Place the load cell an a level stable surface.");

  Serial.println("Remove any load applied to the load cell.");

  Serial.println("Send 't' from serial monitor to set the tare offset.");

  boolean \_resume = false;

  while (\_resume == false) {

    LoadCell.update();

    if (Serial.available() > 0) {

      if (Serial.available() > 0) {

        char inByte = Serial.read();

        if (inByte == 't') LoadCell.tareNoDelay();

      }

    }

    if (LoadCell.getTareStatus() == true) {

      Serial.println("Tare complete");

      \_resume = true;

    }

  }

  Serial.println("Now, place your known mass on the loadcell.");

  Serial.println("Then send the weight of this mass (i.e. 100.0) from serial monitor.");

  float known\_mass = 0;

  \_resume = false;

  while (\_resume == false) {

    LoadCell.update();

    if (Serial.available() > 0) {

      known\_mass = Serial.parseFloat();

      if (known\_mass != 0) {

        Serial.print("Known mass is: ");

        Serial.println(known\_mass);

        \_resume = true;

      }

    }

  }

  LoadCell.refreshDataSet(); //refresh the dataset to be sure that the known mass is measured correct

  float newCalibrationValue = LoadCell.getNewCalibration(known\_mass); //get the new calibration value

  Serial.print("New calibration value has been set to: ");

  Serial.print(newCalibrationValue);

  Serial.println(", use this as calibration value (calFactor) in your project sketch.");

  Serial.print("Save this value to EEPROM adress ");

  Serial.print(calVal\_eepromAdress);

  Serial.println("? y/n");

\_resume = false;

  while (\_resume == false) {

    if (Serial.available() > 0) {

      char inByte = Serial.read();

      if (inByte == 'y') {

#if defined(ESP8266)|| defined(ESP32)

        EEPROM.begin(512);

#endif

        EEPROM.put(calVal\_eepromAdress, newCalibrationValue);

#if defined(ESP8266)|| defined(ESP32)

        EEPROM.commit();

#endif

        EEPROM.get(calVal\_eepromAdress, newCalibrationValue);

        Serial.print("Value ");

        Serial.print(newCalibrationValue);

        Serial.print(" saved to EEPROM address: ");

        Serial.println(calVal\_eepromAdress);

        \_resume = true;

   }

      else if (inByte == 'n') {

        Serial.println("Value not saved to EEPROM");

        \_resume = true;

      }

    }

  }

  Serial.println("End calibration");

  Serial.println("\*");

  Serial.println("To re-calibrate, send 'r' from serial monitor.");

  Serial.println("For manual edit of the calibration value, send 'c' from serial monitor.");

  Serial.println("\*");

}

void changeSavedCalFactor() {

  float oldCalibrationValue = LoadCell.getCalFactor();

  boolean \_resume = false;

  Serial.println("\*");

  Serial.print("Current value is: ");

  Serial.println(oldCalibrationValue);

  Serial.println("Now, send the new value from serial monitor, i.e. 696.0");

  float newCalibrationValue;

  while (\_resume == false) {

    if (Serial.available() > 0) {

      newCalibrationValue = Serial.parseFloat();

      if (newCalibrationValue != 0) {

        Serial.print("New calibration value is: ");

        Serial.println(newCalibrationValue);

        LoadCell.setCalFactor(newCalibrationValue);

        \_resume = true;

      }

    }

  }

  \_resume = false;

  Serial.print("Save this value to EEPROM adress ");

  Serial.print(calVal\_eepromAdress);

  Serial.println("? y/n");

  while (\_resume == false) {

    if (Serial.available() > 0) {

      char inByte = Serial.read();

      if (inByte == 'y') {

#if defined(ESP8266)|| defined(ESP32)

        EEPROM.begin(512);

#endif

        EEPROM.put(calVal\_eepromAdress, newCalibrationValue);

#if defined(ESP8266)|| defined(ESP32)

        EEPROM.commit();

#endif

        EEPROM.get(calVal\_eepromAdress, newCalibrationValue);

        Serial.print("Value ");

        Serial.print(newCalibrationValue);

        Serial.print(" saved to EEPROM address: ");

        Serial.println(calVal\_eepromAdress);

        \_resume = true;

      }

      else if (inByte == 'n') {

        Serial.println("Value not saved to EEPROM");

        \_resume = true;

      }

    }

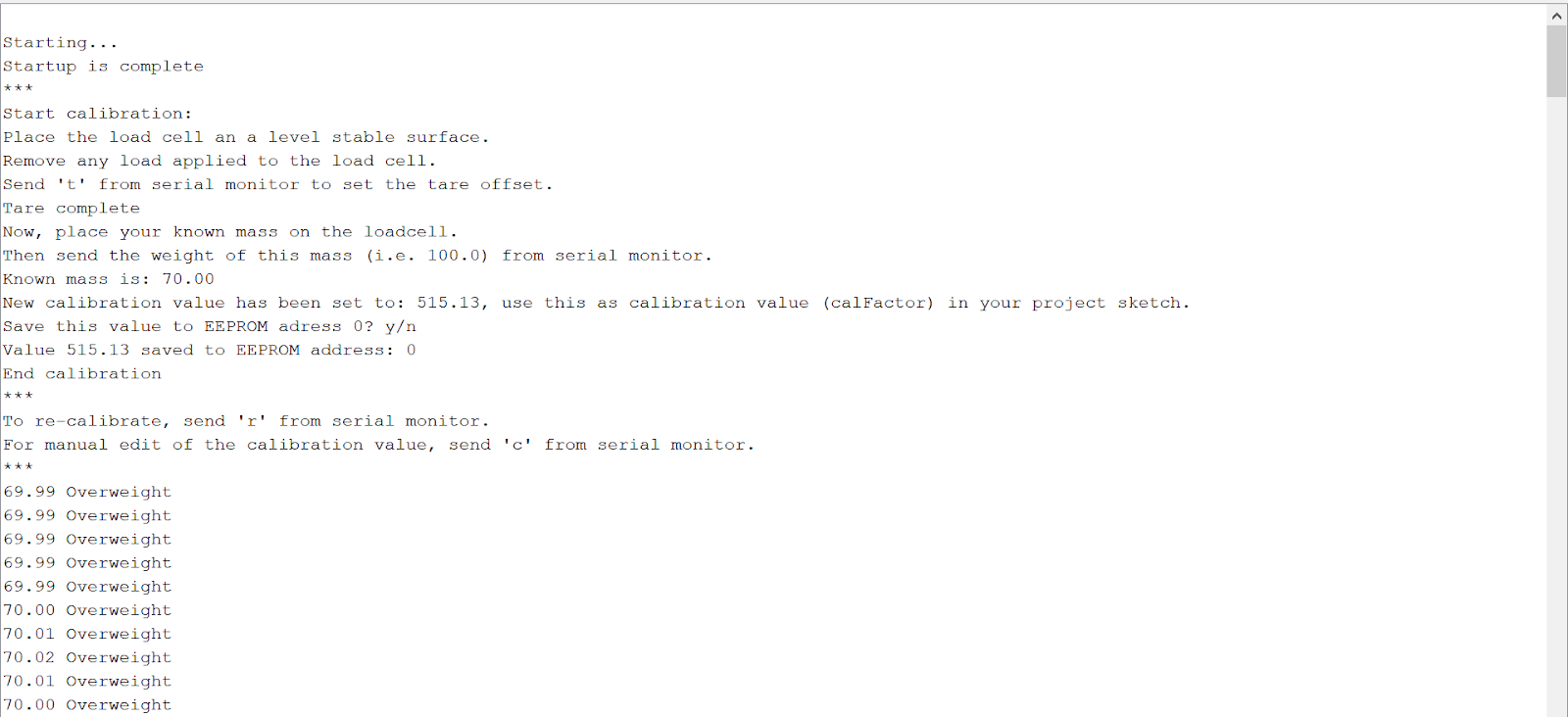
  }

  Serial.println("End change calibration value");

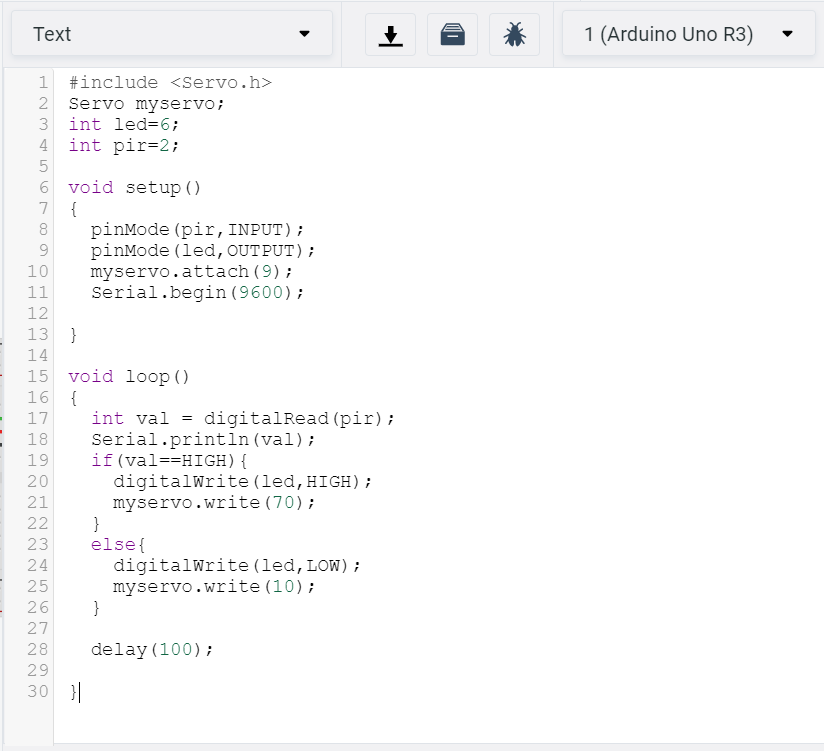
  Serial.println("\*");

}

**3.1.2 OUTPUT SCREENSHOT:**

****

***3.2 TINKERCAD CODE (SOFTWARE):***

****

**3.3 CODE ANALYSIS**

* Number of Lines: 212 (ARDUINO)

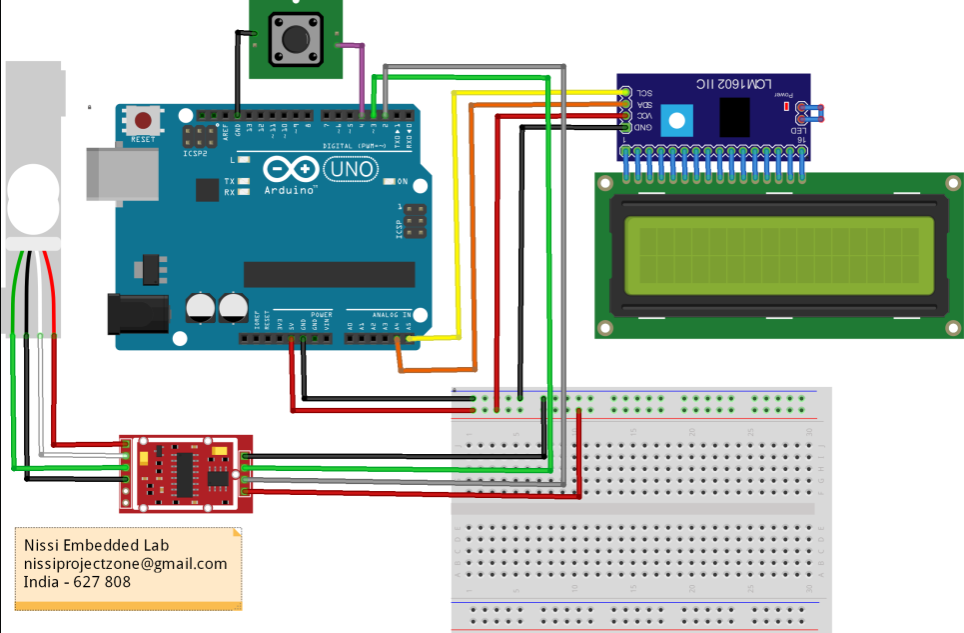
We use this code of program to display the weight and indicate whether or not the input weight exceeds the given threshold, on which case, it displays “OVERWEIGHT” along with the printed weight. Buzzer also beeps with a delay of 1 second.

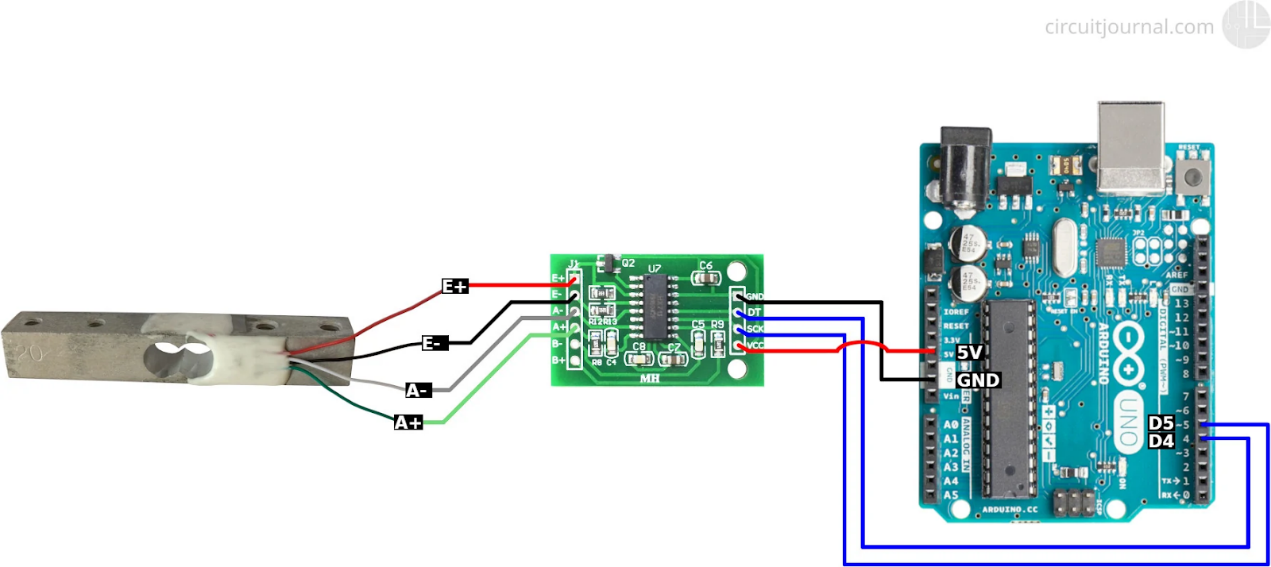
* Number of Lines: 30 (TINKERCAD)

We basically implemented this code to run the PIR sensor. If a person or an object moves in front of the sensor, it detects them and the LED light begins blinking thereby indicating the presence of a person/object.

**3.4 HARDWARE CONNECTIONS**

* Arduino Uno is interfaced with the HX711 module through Ports D4 and D5. The rs, rw and E-pin is connected to P1.2, P1.3, P1.4 respectively and D0-D7 are connected to P2.0-P2.7.
* Attached below is the hardware connections set-up of our project.





**SUMMARY**

Our project mainly entails to describe the study and application of new sensing and data analysis techniques, applied to the determination of space occupancy. A lot of attention has been drawn on automatic detection or in our case occupancy detection to save energy by using various sensor data to automatically control indoor utility such as air conditioning, light, etc. The motion sensor is of IR (Active Infrared) type. IR sensors act as a transceiver, that is, it emits light from the transmitter end and when an obstacle is detected, the reflected light is detected by the receiver end.

The Arduino is programmed in such a way that it will sense the weight of the person or the object and conclude whether or not the threshold limit of the elevator has been surpassed. Since our project only deals with the prototype model and the load cell that we have used has a maximum capacity of 5kg, we’ve set the sample threshold weight to be 50 grams.

Now if the weight of the object exceeds 50g, the LCD module will display “Overweight” and the buzzer will start beeping with a delay of 1000 millisecond (1 second). Another alternative would be to use an LED for the prototype, where the LED is made to blink if the object is overweight.

**CONCLUSION**

The traditional method for monitoring elevator overload is to use gravity sensors. Moreover, to prevent the failure of the gravity sensor, the method of using computer vision to count people in the elevator to determine whether the elevator is overloaded has attracted the attention of many researchers today. Hence in order to count the number of people more accurately while maintaining the safety, a load cell is used which will help us compute the weight of the person entering the elevator. If the weight surpasses a certain maximum limit no more people will be allowed to enter the elevator.

Thus, the project entitled “OCCUPANCY DETECTION SYSTEM” helps to estimate the weight of the people entering the lift and thereby stipulates if the lift has reached its maximum carrying capacity.

**FUTURE ENHANCEMENT**

* The device should be installed at a narrow entrance suitable for only one person to pass through at a given time.
* An uninterruptible power supply should be introduced to the system to serve as a backup power supply. In the near future, some institutions that deem it necessary to monitor their crowd may no longer rely solely on human auditors and unsophisticated counter systems to tally the number of visitors.
* Also the object counting is not limited to the entry/exit point of a lift but has a wide range of applications that provide information to management on the volume and flow of people throughout a location.
* The Counter can be used to maximize the efficiency and effectiveness of a place, floor area and sales potential of an organization.
* The circuit may also be enhanced with a wide counting range of above three digits by modifying the software section of the system.
* It can also be enhanced for long and accurate sensing range using a laser torch instead of IR transmission circuit.
* Thus, the circuit can be used to monitor visitor flow in an effective manner, where the visitors can be counted and controlled.

**REFERENCES**

1. <https://github.com/olkal/HX711_ADC>
2. <https://circuitjournal.com/four-wire-load-cell-with-HX711>
3. <https://mschoeffler.com/2017/12/04/arduino-tutorial-hx711-load-cell-amplifier-weight-sensor-module-lcm1602-iic-v1-lcd/>
4. <https://how2electronics.com/weighing-machine-arduino-load-cell-hx711/>
5. <https://www.electroniclinic.com/hx711-load-cell-or-strain-gauge-and-arduino/>