NATIONAL INSTITUTE OF TECHNOLOGY AGARTALA

ELECTRICAL AND ELECTRONICS

INSTRUMENTATION MEASUREMENT PROJECT

NAME: DIGANTA DAS

ENROLLMENT NUMBER:21UEI036

BRANCH: ELECTRONICS AND ISNTRUMENTATION ENGINEERING

3rd YEAR

5th SEMESTER.

TOPIC OF THE PROJECT

BUILD A STATIC FORCE MEASURING SYSTEM USING ARDUINO AND STRAIN GAUGE.

TABLE OF CONTENT

1. INTRODUCTION

Force measurement is a fundamental aspect of scientific research and engineering applications, essential for understanding and characterizing the physical world. Whether in material testing, structural analysis, or product design, the ability to accurately measure and quantify forces is critical. A static force measuring system is designed to assess forces at rest, offering precise data for various scientific and engineering endeavors.

This introduction outlines the significance and purpose of building a static force measuring system using an experimental setup. It highlights the importance of force measurement, introduces the key components of the experimental setup, and provides a glimpse into the objectives of this project.

2. USES OF THIS PROJECT

A static force measuring system, typically used in experimental setups, can serve various purposes across different fields of science and engineering. The exact applications may vary based on the specific requirements of your project and the experimental setup you're working with. Here are some common uses for a static force measuring system in experimental setups:

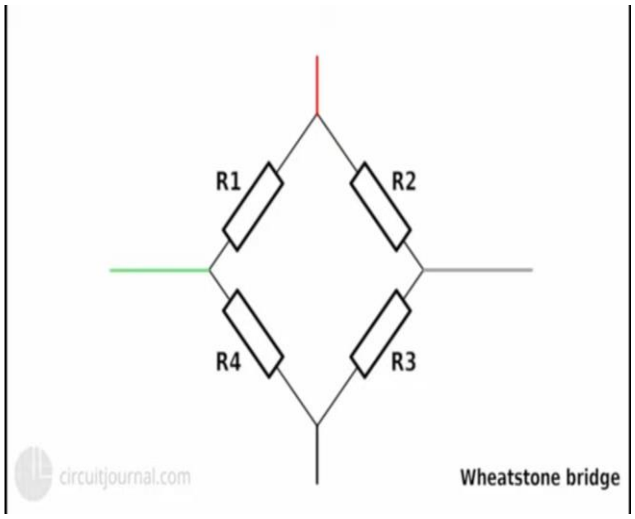
1. **Material Testing:**

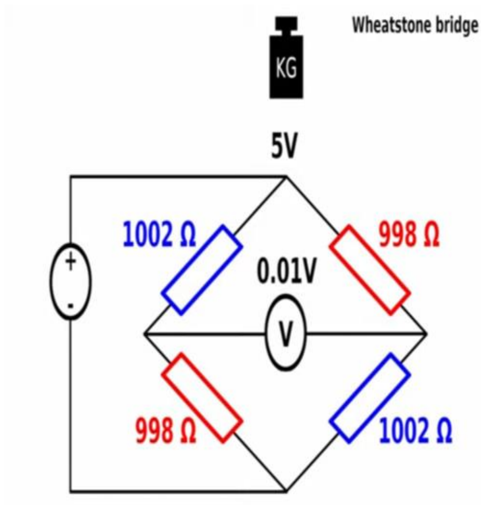
- Determine material properties such as tensile strength, compressive strength, and elasticity.

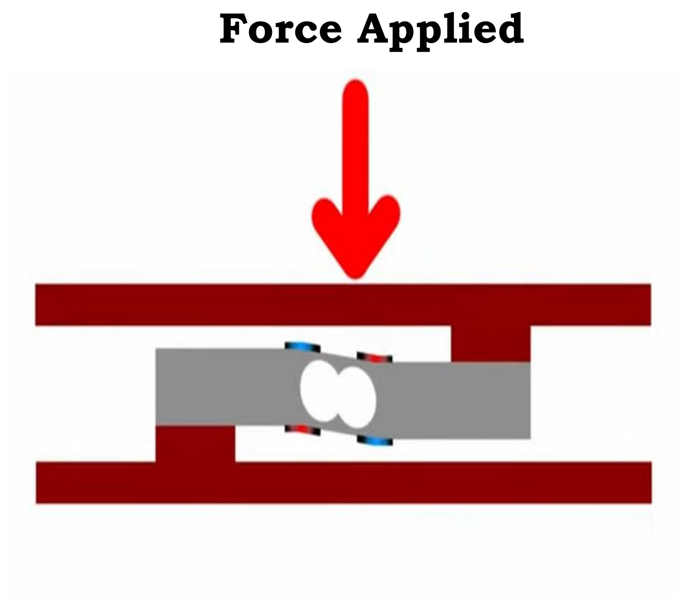
2. **Structural Analysis:**
    - Evaluate the stability and load-bearing capacity of structures.
    - Analyze the effects of forces on bridges, buildings, and other infrastructure.
3. **Product Testing and Quality Control:**
    - Verify the performance and durability of components.
    - Identify weaknesses or defects in manufactured goods.
4. **Geotechnical Engineering:**
    - Measure soil properties and stability.
    - Assess the impact of static loads on the ground.
5. **Aerospace and Automotive Testing:**
    - Assess the performance of aircraft components, landing gear, and vehicle parts under static loads.
    - Determine stress and strain in materials used in aerospace and automotive engineering.
6. **Robotics and Automation:**
    - Ensure safety and control in automated manufacturing processes.
7. **Environmental Monitoring:**
    - Monitor and analyze the forces exerted by environmental factors (e.g., wind, waves, and water currents).
    - Evaluate the effects of static forces on structures in harsh environments.
8. **Research and Development:**
    - Develop and test prototypes of new products and devices.
    - Validate theoretical models and simulations.

3. WORKING PRINCIPLE

Building a static force measuring system using an experimental setup involves designing a device or system that can accurately measure forces acting on an object without the need for dynamic or moving parts. Such a system is often used in scientific research, engineering, and quality control applications. The working principle of this project typically involves the following key components and steps:





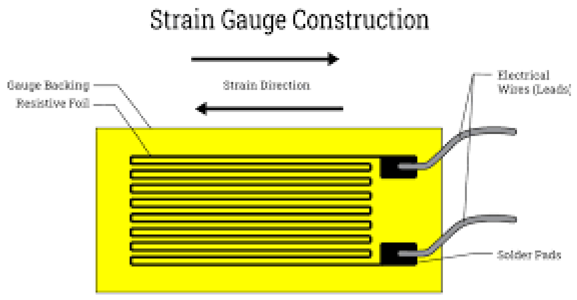


It works on the principle of the **Wheatstone Bridge**.

As tension in strain gauge increases resistance increases, and as it is compressed resistance decreases.

1. A load sensor or transducer. This device converts the applied force into an electrical signal. Common types of load sensors include strain gauges, load cells, and piezoelectric sensors.

2. Strain Gauges: If you are using strain gauges, they work on the principle that the electrical resistance of a material changes when it is subjected to mechanical deformation. When a force is applied to the object, the strain gauges attached to it deform, causing a change in their resistance.



3. Wheatstone Bridge Circuit: To measure the change in resistance, a Wheatstone bridge circuit is often used. This circuit balances the resistances in such a way that a small change in resistance due to applied force leads to a detectable change in voltage across the bridge.

4. Signal Conditioning: The output of the Wheatstone bridge is a voltage signal that corresponds to the applied force. This signal may be very small and noisy, so it needs to be conditioned and amplified for accurate measurement. Signal conditioning involves filtering, amplification, and sometimes digitization of the signal.

5. Data Acquisition: The conditioned signal is then passed to a data acquisition system. This system may consist of analog-to-digital converters (ADCs) to convert the analog voltage signal into digital data that a computer or microcontroller can process.

6. Calibration: The system must be calibrated to relate the measured voltage to the actual force. This is done by applying known forces to the sensor and recording the corresponding voltage readings.

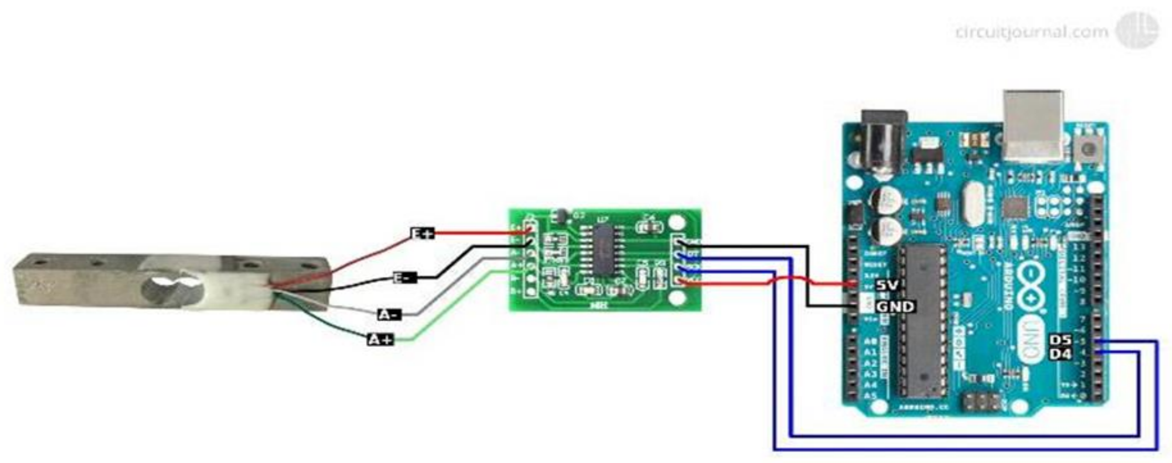
7. Data Processing and Display: The digitized force data is processed by a microcontroller or computer. It can be displayed on a screen or saved for further analysis. You may also add features like data logging and real-time monitoring.

8. Application-Specific Features: Depending on your specific project, you may need to incorporate additional features, such as data logging, alarms for threshold force levels, or communication interfaces for remote monitoring.

It works on the principle of Wheatstone bridge.

As tension in the strain gauge increases resistance increases, and as it is compressed resistance decreases.

4. CIRCUIT DIAGRAM



Link

5. CONNECTIONS

1. Connect the red wire to the E+ and the black wire to the E- E-output of the HX711 module.

We chose the red and black wire pair to be the power wires of the load cell. E+ and E- are the sensor power outputs of the HX711 module.

The polarity doesn't matter. We picked red to be the positive side and black to be the negative side to make it follow a common convention. Switching up red and black will only invert the calibration parameter in the software.

2. Connect the green wire to the A+ and the white one to the A- inputs of the HX711 module.

A+ and A- are the measurement inputs of the HX711 module. Like with the power wires, the polarity is not important. You just need to recalibrate the software if you switch them up.

3. Connect the GND of the HX711 module to the Arduino GND and VCC to the Arduino 5V pin.

HX711 also works with 3.3V. If you have some other microcontroller that runs on 3.3V, you can use 3.3V instead of 5V.

4. Connect the DT and SCK of the HX711 module to any of the Arduino digital IO pins.

In the schematic, I used pins 4 and 5, since those are the default pins for the examples of the "HX711_ADC" library. If you want to use interrupts to update scale data, then you should connect the DT output to an interrupt-enabled pin of the Arduino. For Uno/Nano, those are pins 2 and 3.

5. COMPONENTS REQUIRED

- A four-wire load cell with the HX711 module

- An Arduino Uno or Uno-compatible board

- Jumper wires to connect the HX711 module to your Arduino

6. CODE

```
#include <HX711_ADC.h>

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

#include <EEPROM.h>

#endif

//pins:

const int HX711_dout = 4; //mcu > HX711 dout pin

const int HX711_sck = 5; //mcu > HX711 stick 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();

  //LoadCell.setReverseOutput(); //uncomment to turn a negative output value to positive

  unsigned long stabilizing time = 2000; // precision 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

}

void loop() {

  static boolean newDataReady = 0;

  const int serialPrintInterval = 0; //increase value to slow down serial print activity

  // check for new data/start next conversion:

  if (LoadCell.update()) newDataReady = true;

  // get smoothed value from the dataset:

  if (newDataReady) {

    if (millis() > t + serialPrintInterval) {

      float i = LoadCell.getData();
      Serial.print("Load_cell output val: ");
      Serial.println(i);
      newDataReady = 0;
      t = millis();
    }
  }
  // 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("***");
}
```

7.CONCLUSION

In conclusion, utilizing strain gauges for force measurement is a precise and versatile method, offering a range of applications across engineering and science. These devices are capable of converting mechanical force into measurable electrical signals with high sensitivity and accuracy. By monitoring the strain-induced changes in resistance, strain gauges provide valuable data for structural analysis, material testing, and load monitoring. Their ability to deliver real-time feedback and compatibility with modern data acquisition systems makes them indispensable tools in fields such as aerospace, civil engineering, and industrial quality control. Strain gauges continue to play a crucial role in advancing our understanding of force dynamics and ensuring the safety and reliability of various structures and components.