INTERNET OF THINGS

BECE351E

TF2

FORCE MEASUREMENT USING SK6511 LOADCELL SENSOR AND JY-S60 WEIGHT TRANSMITTER

FACULTY:

Dr. Edward Jero Sam Jeeva Raj

TEAM MEMBERS:

21BPS1278-Shashank Reddy

21BAI1759-Kalyan Chakravarthy

21BRS1022-Hanuma Vihari

21BPS1504-Sompalli Rahul

21BPS1547-Rakesh Reddy

FORCE MEASUREMENT USING SK6511 LOADCELL SENSOR AND JY-S60 WEIGHT TRANSMITTER

Abstract:

The Internet of Things (IoT) is a network of interconnected computing devices that enables data collection, processing, and transmission without human intervention. Smart agriculture is necessary, and this project offers a Smart Weighing System as an example of how agriculture can be automated. The system allows farmers to weigh their products by packing them into crates. It improves upon the standard weighing machine by using sensors and logics. A smart weighing machine uses sensors to weigh items and sends messages to track containers using GSM modems, turning a conventional weighing machine into a smart device.

Resilience is crucial when building a bridge, as its ability to uphold during load transfer depends on it. Bridge loading is dynamic and subject to change, making it challenging to estimate the bridge's lifetime. An automatic load detector is required to estimate overloading and evaluate its impacts on the bridge. This study developed an automatic load detector using a load cell sensor based on Arduino. The detector has a buzzer switch with a warning sound and an I2C LCD display mechanism that shows the load on a bridge and notifies when it is overloaded. The average error rate in the experimental data is 4.67%, making the sensor system suitable for assessing bridge deterioration. The detector is efficient at evaluating dynamic loading conditions to prevent bridge damage.

Remote monitoring of elderly patients' activities and daily lives (ADLs) in the event of a fall is a primary goal of telemedicine. AI methods, such as machine and deep learning models, and IoT have been used to automate diagnosis processes. Real-time detection of senior patient fall accidents is one of their many uses. Various sensor types, including gyroscopes and accelerometers in smartwatches, are examined for creating viable FDSs. Lightweight deep models and IoT-enabled smart devices are suggested as potential solutions for accurate fall detection in the elderly.

Falls are a frequent reason for elderly individuals to need medical attention. Technologies using webcams to monitor elderly activities are expensive and only useful indoors. Wristwatch-style wireless emergency transmitters limit mobility and generate false alarms. This research presents a reliable and cost-effective fall detection system using an accelerometer and gyroscope to measure acceleration and body tilt angle, respectively. The system sends an SMS alert to relevant authorities and has a low implementation cost. The sensitivity and specificity of the fall detection and alert system are 95% and 90%, respectively.

The extensive use of load cells are accurate sensors for measuring force and torque in various industries. Load cells are widely employed for weight measurement, including in food, automotive, and animal weighing. They provide compression force feedback in robotic grippers to prevent damage or premature release. Load cells also assist walking robots by sensing compression forces for control systems. Industrial machines utilize load cell-equipped rods, beams, wheels, and bars to measure acting forces. They can indirectly estimate tank volume or level based on overall weight. The abstract discusses load cell theory, application, and required electronics, presenting a specific example of a 3-ring spherical load cell for quantifying compression forces on stored and transported fruit.

Keywords: IoT, Smart Agriculture, Weighing System, Cloud Computing, Fall detection, Wearable sensors.

Introduction:

The aim of this project report is to present a detailed analysis of the implementation and results of a force measurement system utilizing the ESP32 microcontroller, the SK6511 load cell sensor, and the JY-S60 weight transmitter. This project focuses on developing a reliable and accurate force measurement solution for various applications where real-time data acquisition and processing are crucial.

Force measurement plays a vital role in numerous fields, including industrial automation, robotics, aerospace, and material testing. Traditionally, strain gauge-based load cells have been widely employed for measuring force, but they often require complex signal conditioning circuits and dedicated instrumentation. However, advancements in microcontroller technology have paved the way for more cost-effective and versatile force measurement systems.

The ESP32 microcontroller, known for its powerful processing capabilities, low power consumption, and built-in Wi-Fi and Bluetooth connectivity, serves as the central unit for this project. It acts as an interface between the load cell sensor and the weight transmitter, acquiring force data from the load cell and transmitting it wirelessly to a remote device for further analysis.

The SK6511 load cell sensor employed in this project is a high-precision, strain gauge-based transducer designed to measure force or weight. It offers excellent linearity, stability, and sensitivity, making it ideal for accurate force measurements. The load cell sensor converts the applied force into an electrical signal that is proportional to the force being exerted.

To ensure reliable and real-time data transmission, the JY-S60 weight transmitter is used in conjunction with the ESP32 microcontroller. The weight transmitter receives the force data from the microcontroller and wirelessly transmits it to a receiver or a remote device, allowing for convenient monitoring and analysis of the measured force.

The key objectives of this project include:

1. Integrating the ESP32 microcontroller with the SK6511 load cell sensor to accurately measure and acquire force data.
2. Establishing communication between the microcontroller and the JY-S60 weight transmitter for seamless wireless transmission of force data.
3. Developing a user-friendly interface for real-time monitoring and analysis of the measured force on a remote device.
4. Validating the accuracy and reliability of the force measurement system through experimental testing and comparison with established reference measurements.

Throughout this project report, the system design, hardware components, software implementation, and experimental results will be thoroughly discussed and analysed. The aim is to provide a comprehensive understanding of the force measurement system using the ESP32 microcontroller, SK6511 load cell sensor, and JY-S60 weight transmitter, while highlighting its potential applications and future improvements.

By accomplishing the objectives outlined above, this project intends to contribute to the field of force measurement by providing a cost-effective, versatile, and accurate solution that can be utilized in various industries and research domains.

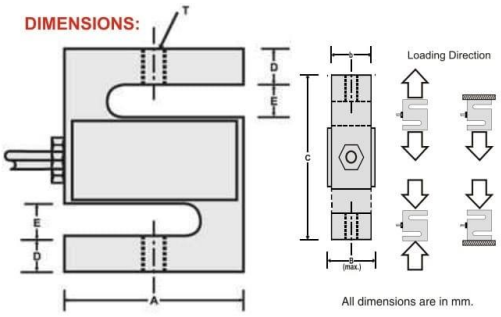
Load Cell Sensor

Features

- Alloy Steel Construction
- Glue Sealing And Welding to IP67 & IP68
- Easy To Install and use for both tensile And Compression application

Applications

Suitable for handing scale, vibratory feeding equipment, tension testing machine, tank weighing, hopper weighing etc.



CAPACITY Kgf.	A	B (max)	b	C	T	D	E	Torque m. kg.	Cable Mtrs
2/5/10	50	20	12.5	55	M6 x 1.000	10	9	0.25	5
20 to 100	50	20	12.5	62	M6 x 1.000	10	9	0.25	5
150 TO 1000	50	25	19	62	M12 x 1.25	10	9	0.50	5

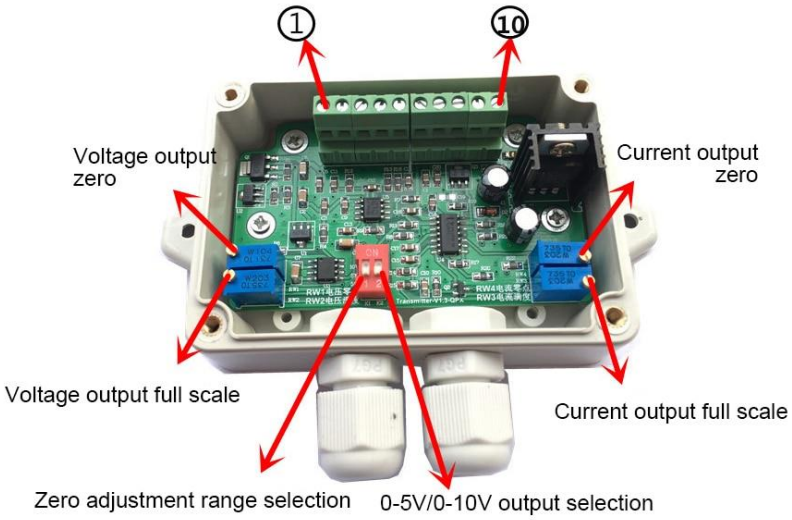
SPECIFICATION

RATED LOAD (Kg.) 2, 5, 10, 20, 50, 100, 200, 300, 500, 750, 1000			
Precision	C2/C3	Insulation Resistance (MΩ)	≥20000(50VDC)
Composition Error	0.03/0.02	Excitation Voltage (V)	5 ~15 (DC)
Rated Output (mv/v)	3.0 ± 0.2	Compensated Temp. Range (°C)	-10~+ 50
Non-Linearity (%FS)	0.025	Use Temp. Range (°C)	-20~+60
Hysteresis (%FS)	0.02	Temp. Effect On Zero (%FS/10°C)	0.02
Repeatability (%FS)	0.02/0.01	Temp. Effect On Span (%FS/10°C)	0.02
Creep (%F.S /30min)	0.03/0.02	Safe Overload (%FS)	150
Zero Balance (%FS)	± 1.0	Ultimate Overload (%FS)	250
Input Resistance (Ω)	400 ± 10	Defend Grade	IP67
Output Resistance (Ω)	350 ± 2	Cable	5 mtr

JY-S60 Weight Transmitter

port definition:

serial number	1	2	3	4	5	6	7	8	9	10
definition	sensor side					Power supply and signal output				
	excitation just	Signal just.	Signal burden	excitation burden	shield Wire	shield Wire	public	Voltage output	current output	power supply just



Literature Survey:

Review 1

Title: IOT based Smart weighing system for Crate in Agriculture

Authors: Pravin Sonsare

Publisher: International Journal of Computer Sciences and Engineering (2018)

Summary:

The Internet of Things (IoT) is a network of physical objects embedded with electronics, software, sensors, and network connectivity, enabling remote sensing and control of objects. This allows for more direct integration of the physical world into computer-based systems, improving efficiency, accuracy, and economic benefits. A home IoT system integrates sensors into a network, completing daily functionality without human intervention. Design criteria include selecting specific sensors, choosing networks and protocols, determining processing capability, ensuring minimal power consumption, and providing additional interfaces for control mechanisms. An example of an IoT system is the Smart weighing system, which monitors goods weight automatically and alerts users when the crate is about to fill. The system is flexible enough to be accessed via a mobile app, allowing users to check the status of goods and communicate with the system.

The project aims to implement mobile applications for farming that enable simple and efficient data input during daily activities. The scenarios-oriented user interface will enable farmers to have a sequenced order of activities, enhancing the existing system with a recommended plan for next day activities. This will lead to significant increases in farming efficiency, more efficient use of resources, and savings in administration work. Additionally, the platform will enable the collection of data from sensors and other devices through wireless communication technologies. This data will provide new possibilities for analysis and decision support, as well as better possibilities for requesting additional data for electronic reporting.

Advanced decision support for farming will be implemented, allowing for timely and specific consulting services on the spot, more efficient use of resources, and increased collaboration within industries and communities. The platform will also offer consulting services in the cloud for farming, allowing for consulting in general or areas with high expertise needed.

The project will also integrate individual elements into the platform and create open standards for integration, published under public domain or creative-commons license. This will allow for collaboration and large support within industries and communities. The expected outcomes include accurate load measurements, cost efficiency, time savings, flexibility, and data management.

Load cells are passive transducers or sensors that convert applied force into electrical signals, such as fluid pressure, elasticity, or magnetostriction. The HX711 Weighing Sensor Module amplifies the low electric output of load cells, which is then fed into the Arduino to derive the weight. The Arduino UNO is a microcontroller board based on the ATmega328P, with 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. The Arduino UNO is easy to tinker with and can be replaced with a few dollars. Cloud service is another option for automating weighing systems, as it allows for secure data processing without a hard drive or dedicated server. This option is ideal for operations that need to compile data from remote locations.

Review 2

Title: Automatic Load Detector Design to Determine the Strength of Pedestrian Bridges Using Load Cell Sensor Based on Arduino.

Author: Riyanti, Kurnia & Kakaravada, Ismail & Ali Ahmed, Abdussalam

Publication: Indonesian Journal of Electronics Electromedical Engineering and Medical Informatics (2022)

Summary:

Indonesia relies heavily on bridges for inland transportation and economic support. Bridge construction materials depend on the user's capacity and the number and load of the bridge. Bridges must meet various standards, including resistance to withstand the burden of humans

and vehicles, and strain conditions. Damages to bridges can occur due to underground construction, soil conditions, or external factors like temperature, pressure/strain, and earthquakes. Monitoring of bridge performance, maximizing the lifespan of the bridge, and preventing damage due to load overload is crucial.

In this study, an automatic detector was introduced to determine the strength of a bridge under dynamic loading conditions. The load cell detector uses sensors with a maximum capacity of 25 kg, mounted under the bridge surface in a tiny form. The study aims to detect structural deformations caused by normal operations or environmental impacts, such as bridge age, daily traffic, and location.

Various studies have been conducted on load test evaluation, vibration signal analysis, and dynamic loading evaluation. Some researchers are still focusing on load cell sensors for testing loads and calculating weights, but there is still a gap in dynamic loading evaluation and estimation of bridge life time. The current research focuses on developing load cell sensors using Arduino systems to evaluate the load distribution on the structure.

This study uses four HX711 load cell sensors with a maximum weight specification of 25 kg as a load counter sensor, an Arduino UNO R3 microcontroller as a load value processor, an LCD 16x2 I2C display, and a buzzer as an alarm if the miniature bridge's load exceeds the 25 kg limit. Two types of loads were tested on miniature bridges for pedestrians: static load and walking load. The static load test detects which load cell reads the load and checks the accuracy of reading results. The walking load test checks the accuracy of load cell readings and the tendency of the load to lead to any load cell. The HX711 load cell reading sensor requires an input voltage of +5 VDC and ground, and outputs digital data connected to digital pins 2, 3, 4 and 5.

The LCD screen displays the load reading when less than 25 kilograms, and a buzzer sounds if the load reading is more than 25 kilograms. The installation of a load cell must be careful, with the part connected to the microcontroller and the part connected to the load having an arrow down indicating proper reading.

The automatic load detector successfully evaluates bridge strength at static and dynamic loading conditions, but faces 4.67% error in dynamic load detection. The Arduino-based load cell sensor is more effective for estimating bridge life span.

Review 3

Title: Survey of IoT-Based Fall Detection for Aiding Elderly Care: Sensors, Methods, Challenges and Future Trends

Author: Karar, Mohamed & Shehata, Hazem & Reyad, Omar

Publication: Applied Sciences (2022)

Summary:

The World Health Organization (WHO) reports that approximately 684,000 disastrous falls occur annually, with the majority of victims being individuals over the age of 60. Falls are a major public health concern for the elderly, with over 80% in low-income and middle-income countries. The injuries sustained by falls have far-reaching effects on families, healthcare institutions, and society in general. Fall detection technology in medical warning systems is crucial and life-saving. These devices use alert systems technology to identify and provide emergency assistance to seniors who are prone to falls.

Accelerometers, low-power radio wave technology sensors, are used in fall detection systems to continuously monitor the user's movements. Some fall detection systems have built-in tri-axial accelerometers that use Biosensor's patented algorithms. These devices can assess a person's body position, physical activity, and the smoothness of movements, determining whether a user has fallen. If these variables are in the danger zone and a fall has happened, the smart device will automatically activate an emergency fall alarm and contact emergency response agents for assistance.

Personal emergency response systems (PERS) are a commercial approach for preventing falls, but they are rendered worthless if the user is completely unconscious or unable to reach the button. Passive monitoring approaches have been proposed to identify falls due to the problems connected with PERS systems. Several options are currently available, most of which are wearable devices, cameras, microphones, and pressure sensors implanted beneath the flooring.

This article presents the first survey of IoT-based fall detection systems, including different wearable and non-wearable sensor types, machine learning, and deep learning detection algorithms. The article also discusses the challenges and future trends of fall detection systems, such as the availability of public datasets of falls for senior people and the development of new lightweight deep models for accurate fall detection with minimal hardware resources.

This article presents various fall detection algorithms and systems based on IoT technology, primarily using artificial neural networks for automated decision-making. Smartphones and smart watches offer an opportunity to develop wearable FDSs using built-in hardware

resources and wireless sensors. Evaluating fall datasets and using artificial intelligence techniques can enhance the robustness of these systems. However, a pre-processing stage, such as thresholding, is needed to reduce error possibilities and minimize computational and power costs. Deep learning techniques and convolutional neural network architectures are emerging as fall detection algorithms, offering advantages over classical methods. Lightweight deep models are also proposed for mobile applications, potentially presenting a new version of smart IoT-based classifiers for elderly patients' fall accidents.

Review 4

Title: Smart Fall Detection System for Elderly People with IOT and Sensors

Author: Dishari Sengupta, Subhadip Ray, Progati Biswas, Subham Rajak, Dr. Debasish Mondal

Publication: Applied Sciences (2021)

Summary:

This article presents various fall detection algorithms and systems based on IoT technology, primarily using artificial neural networks for automated decision-making. Smartphones and smart watches offer an opportunity to develop wearable FDSs using built-in hardware resources and wireless sensors. Evaluating fall datasets and using artificial intelligence techniques can enhance the robustness of these systems. However, a pre-processing stage, such as thresholding, is needed to reduce error possibilities and minimize computational and power costs. Deep learning techniques and convolutional neural network architectures are emerging as fall detection algorithms, offering advantages over classical methods. Lightweight deep models are also proposed for mobile applications, potentially presenting a new version of smart IoT-based classifiers for elderly patients' fall accidents.

IoT (Internet of Things) is an advanced automation and analytics system that utilizes networking, sensing, big data, and artificial intelligence technology to deliver complete systems for products or services. These systems offer greater transparency, control, and performance across industries, enhancing data collection, automation, operations, and more through smart devices and powerful enabling technology. IoT systems enable deeper automation, analysis, and integration within a system, improving the reach of these areas and their accuracy. Key features of IoT include artificial intelligence, connectivity, sensors, active engagement, and small device use.

IoT offers numerous advantages, such as improved customer engagement, technology optimization, reduced waste, and enhanced data collection. However, it also presents challenges such as security, privacy, complexity, flexibility, and compliance. IoT software addresses these issues through platforms, embedded systems, partner systems, and middleware.

Data collection software manages sensing, measurements, light data filtering, light data security, and aggregation of data. Device integration software supports integration and binds all system devices to create the body of the IoT system. Real-time analytics applications analyze information based on various settings and designs to perform automation-related tasks or provide industry data. Application and process extension applications extend the reach of existing systems and software to allow a wider, more effective system.

IoT primarily exploits standard protocols and networking technologies, but major enabling technologies and protocols of IoT include RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and WiFi-Direct. These technologies support the specific networking functionality needed in an IoT system, compared to a standard uniform network of common systems.

Low-Energy Wireless replaces power-hungry IoT systems, reducing consumption and device life. Radio Protocols like ZigBee, Z-Wave, and Thread create low-rate private area networks with high throughput. LTE-A upgrades LTE technology by increasing coverage, reducing latency, and increasing throughput, enhancing IoT applications in vehicles and UAVs. WiFi-Direct eliminates access points, enabling P2P connections with lower latency and maintaining speed and throughput.

The proposed device uses fall-feature parameters of 6-axes acceleration to detect falls with sensitivity, specificity, and accuracy over 95%. The parameters are adjusted using a simple threshold and can be implemented into an 8051-based microcontroller with 128 Kbyte ROM. The algorithms are simple and can be tested in real-time if implemented in an embedded system. The results demonstrate a reduction in computing effort and resources compared to using all events applied.

Review 5

Title: Load cells in force sensing analysis - Theory and a novel application

Author: Müller, Ivan & Machado de Brito, Renato & Pereira, Carlos & Brusamarello, Valner

Publication: Instrumentation & Measurement Magazine, IEEE (2010)

Summary:

Load cells are essential sensors for detecting and measuring force and torque in various fields, such as weighing food, vehicles, and animals. They are used in robotic arm grippers to provide compression force feedback, and in industrial machinery to measure the forces exerted on rods, beams, wheels, and bars. Load cells can also monitor the total weight of tanks and lift units to prevent overload.

A load cell is a device with various components, including a strain gauge, which is a thin foil resistor bonded to elastic materials like metals. The strain gauge resistance changes according to the deformation of the spring element, which is a result of the Poisson effect. Spring elements with Poisson effects are used with strain gauges that vary in electrical resistance when under stress. The intensity of the electrical resistance variation in strain gauges is proportional to the intensity of the applied force.

The spring element is connected to the object applying the force and has a longitudinal and transversal Poisson effect depending on its material composition. Spring elements are typically made of aluminium or steel and must be within the elastic range of the material to respond linearly. If overloaded beyond the elastic limit, they can suffer permanent damage. Load cells are crucial for various applications, including weighing food, vehicles, and animals, and in industrial machinery, monitoring tank volume and lifting units.

The basic signal conditioning of data from strain gauges involves amplification and excitation of the analog signal from load cell data, filtering, AD modulation, demodulation, and dynamic compensation. The Wheatstone bridge is necessary for amplification, which requires high input impedance, high differential gain, high common-mode and power supply rejection ratio, low drift, low offset, and low input bias currents. Integrated instrumentation amplifiers can be used since they have internal trimmed resistors for a better match and increase in overall performance. Filtering prevents aliasing and removes noise outside the signal's useful band. Noise is introduced in load cell systems by external or internal electromagnetic sources, and a low pass filter is usually employed for near static force measurement results.

Digital signal conditioning is gradually being replaced with analog conditioning, especially at the linearization process. In the digital world, there is no drift, component tolerance, or aging, but the signals will always be analog, and thus, the front end (amplification and filtering) will be also. Ideally, all the ADC dynamic range should be used to take advantage of its full resolution. A low noise instrumentation amplifier and filtering are needed.

After the correct conditioning, the signal is ready to be digitized. Force measurement demands low rate, low noise, and high-resolution quantification. For most applications of non-dynamic measurements, the sampling time is at long intervals. To diminish noise, averages of four or more acquisitions are performed before data are used. Multi-slope or sigma-delta ADCs with resolutions greater than 16 bits and sample rates from 10-1000 S/s are quite adequate.

There are several different types and shapes of strain gauge-based load cells, including shear beams, bending beams, buttons, rings, and canisters. They are capable of sensing compression and traction forces and can be used directly or indirectly in several different applications. For multidimensional applications, some commercial load cells can capture

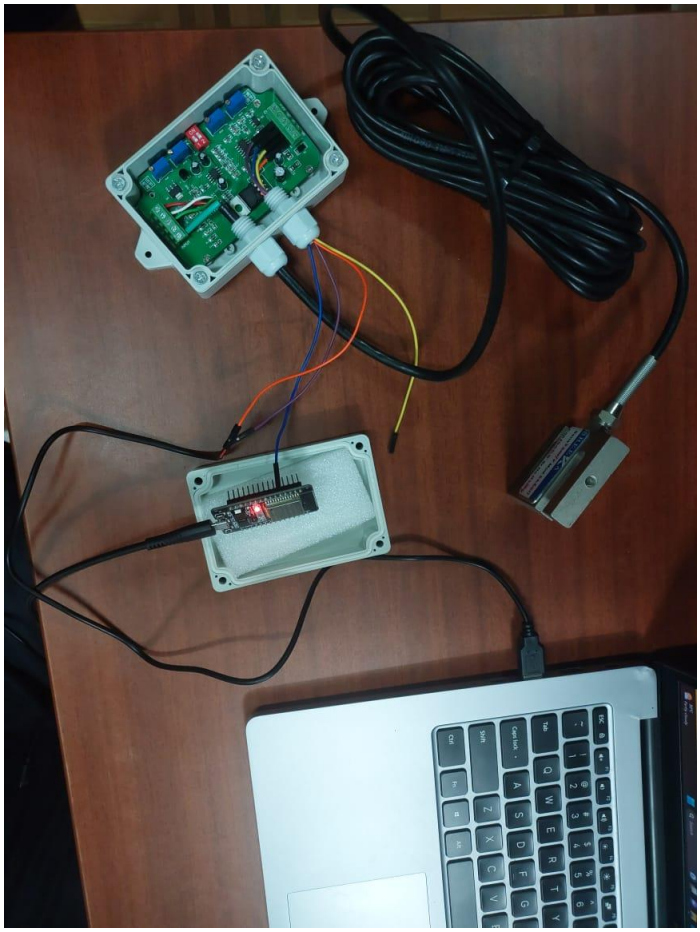
forces along one axis by means of a central rod, equipped with strategically positioned strain gauges. Some types are more sophisticated and capture the three axes' forces and their angular moments with a total of six components.

A new sphere-shape load cell design has been developed to sense all force components that act at only one point. If the forces come from different directions, a combination of load cells can be used. If only three axis compression/traction forces need to be found, a different load cell configuration is proposed. The device is composed of a combination of 3-ring load cells, which allows for maximum bending and independence for each axis.

The study presents load cell theory and a versatile measurement system using a three-ring spherical load cell. The system addresses internal and external curvature issues, enabling post-harvest processing of apples and oranges. C++ software is developed to display compression forces. Future work includes developing a wireless sensor-based device for cooperative communication, enabling more realistic experiments in real-fruit post-harvest systems.

Methodology:

Architecture:



1. The ESP32 initializes the necessary interfaces and pins for communication with the weight transmitter and load cell sensor.
2. The ESP32 continuously reads the output signal from the weight transmitter, which represents the measured force. This can be done using an ADC or a suitable digital communication interface.
3. If an amplifier is used, the output signal from the load cell sensor is amplified before being connected to the weight transmitter or ESP32.
4. The ESP32 may perform any necessary calculations or conversions on the measured force, such as unit conversion or calibration adjustments, depending on the specific application requirements.
5. The ESP32 can process and store the force measurements for further analysis or transmit the data to a remote server or display device for monitoring and visualization purposes.

Sensors:

ESP32 Microcontroller: The ESP32 is a powerful microcontroller with built-in Wi-Fi and Bluetooth capabilities. It will serve as the main control unit for the force measurement system. You can program the ESP32 using the Arduino IDE or the ESP-IDF framework

Load Cell Sensor (SK6511): The SK6511 is a load cell sensor that can accurately measure force or weight. It usually provides analog output in the form of a voltage signal. Connect the load cell sensor to the appropriate pins of the ESP32 for data acquisition.

JY-S60 Weight Transmitter: The JY-S60 weight transmitter is a device that receives weight data from the load cell sensor and transmits it to a receiving device, typically using a communication protocol like RS485 or Modbus. Connect the JY-S60 weight transmitter to the ESP32 using the appropriate communication interface.

Interfaces:

Wiring Connection:

1. Connect the power supply: Connect the positive and negative terminals of the power supply to the corresponding power input terminals of the ESP32, load cell sensor, and weight transmitter. Make sure the voltage supplied is within the operating range of each component.
2. Connect the load cell sensor to the weight transmitter: The SK6511 load cell sensor typically has four wires: red (Excitation+), black (Excitation-), green (Signal+), and white (Signal-). The JY-S60 weight transmitter should have corresponding terminals for these connections. Connect the red wire from the load cell sensor to the Excitation+ terminal of the weight transmitter, the black wire to the Excitation- terminal, the green wire to the Signal+ terminal, and the white wire to the Signal- terminal.
3. Connect the weight transmitter to the ESP32: Depending on the output signal of the JY-S60 weight transmitter, you will need to choose an appropriate interface on the ESP32 to connect to. If the weight transmitter provides an analog output (e.g., 0-10V or 4-20mA), you can use the ESP32's analog-to-digital converter (ADC) pins. Connect the weight transmitter's output signal (e.g., the analog voltage or current) to one of the ADC pins on the ESP32.

4. Connect the ESP32 to the power supply: Connect the positive and negative terminals of the power supply to the corresponding power input terminals on the ESP32. Ensure that the voltage matches the ESP32's requirements.
5. Ground connections: Connect the ground (GND) terminals of the ESP32, load cell sensor, weight transmitter, and power supply together. This helps establish a common reference for all the components.
6. Shielding: If your load cell sensor has a shielding cable, connect the shield wire to the ground (GND) terminal of the weight transmitter or the ESP32. This helps minimize electromagnetic interference.

Communication Protocol:

1. Consult the datasheets and documentation of the SK6511 load cell sensor and JY-S60 weigh transmitter to understand the specific protocol and commands used for communication.
2. The JY-S60 weigh transmitter may have a specific command set for requesting force measurements, calibrating the system, or configuring other parameters. Implement the necessary code to send appropriate commands to the JY-S60 weigh transmitter using weighTransmitterSerial.write() or weighTransmitterSerial.print() functions.

Power supply:

Provide power to the ESP32 and the load cell sensor. Ensure that the voltage levels are compatible with the requirements of each component.

Software setup:

1. Set up the development environment for the ESP32 (e.g., Arduino IDE, ESP-IDF, PlatformIO).
2. Install the necessary libraries or SDKs for ESP32 and load cell communication (if required).
3. Write the firmware code to interface with the load cell sensor and JY-S60 weight transmitter.
4. Configure the communication protocol (UART, SPI, I2C) between the ESP32 and JY-S60.
5. Read the digital output from the JY-S60 weight transmitter.
6. Apply calibration factors, if necessary, to convert the digital output to force or weight values.
7. Implement any additional processing or filtering algorithms as needed.
8. Optionally, integrate Wi-Fi or Bluetooth capabilities to transmit the measured data to other devices or a server.

Testing and calibration:

1. Upload the firmware to the ESP32 and ensure that it can communicate with the load cell sensor and JY-S60 weight transmitter.
2. Perform calibration by applying known weights and verifying the measured values against the expected values.
3. Fine-tune any calibration factors or algorithms to improve the accuracy of the force measurement.

Code (Arduino):

```
// Include the required libraries
#include <Arduino.h>

// Define the pin for the load cell signal
const int loadCellPin = 34;  // D34

// Calibration parameters
const float calibrationFactor = 3.33; // Calibration factor obtained from calibration
void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Configure the Serial Plot feature
  Serial.println("#Labels,Weight");
}

void loop() {
  // Read the load cell signal value
  int loadCellValue = analogRead(loadCellPin);

  // Convert the load cell value to weight using the calibration factor
  float weight = loadCellValue * calibrationFactor;

  // Send the weight data to Serial Plot
  //Serial.print("Force = ");
  Serial.println((weight/100) * 9.8);
  //Serial.println("Newtons");
```

```
  delay(500); // Adjust the delay time as needed
}
```

Sinewave from serial inputs of ESP32:

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

}

void loop() {
  // put your main code here, to run repeatedly:
  int count=0;
  Serial.print(0);
  Serial.print(" ");
  Serial.print(255);
  Serial.print(" ");
  for(int i=0;i<255;i++){
    Serial.println(i);
    delay(200);
  }

}
```

Code (Django):

Views.py

```
from datetime import datetime
import random
from django.http import HttpResponse, JsonResponse
from django.shortcuts import render, redirect, get_object_or_404
from .forms import StudentForm
from .models import StudentModel, readings
from serial import Serial
import struct
```

```
def add_student(request, template_name='student_add.html'):
    form = StudentForm(request.POST or None)
    if form.is_valid():
        form.save()
        return  HttpResponse("ok saved")
        return redirect('Student:student_manage')
    return render(request, template_name, {'form':form})


def student_manage(request,
template_name='student_manage.html'):
    std_data = StudentModel.objects.all()
    data = {}
    data['object_list'] = std_data
    return render(request, template_name, data)


def student_edit(request, pk, template_name='student_add.html'):
    book= get_object_or_404(StudentModel, pk=pk)
    form = StudentForm(request.POST or None, instance=book)
    if form.is_valid():
        form.save()
        return redirect('Student:student_manage')
    return render(request, template_name, {'form':form})

def delete_student(request, pk):
    obj = get_object_or_404(StudentModel, pk=pk)
    obj.delete()
    return redirect('Student:student_manage')



def show_graph(request,template_name='live_graph.html'):
    return render(request,template_name)


def fetch_sensor_values_ajax(request):
    data={}
    float_val=0.0
    if request.is_ajax():
            com_port = request.GET.get('id', None)
            sensor_val=random.random()
            sensor_data=[]
            now=datetime.now()
            ok_date=str(now.strftime('%Y-%m-%d %H:%M:%S'))
            try:
                sr=Serial(com_port,9600)
                st=list(str(sr.readline(),'utf-8'))
```

```
                sr.close()
                hex_str = str(''.join(st[:]))
                sensor_val=str(hex_str)

                if(sensor_val):
                    print(sensor_val)

sensor_data.append(str(sensor_val)+','+ok_date)
                    sensor_val2 = int(float_val)
                    savy = readings(sensor_data =
sensor_val2,time = ok_date)

                else:

sensor_data.append(str(sensor_val)+','+ok_date)
                    sensor_val2 = int(float_val)
                    savy = readings(sensor_data =
sensor_val2,time = ok_date)

            except Exception as e:

sensor_data.append(str(sensor_val)+','+ok_date)
                    sensor_val2 = int(float_val)
                    savy = readings(sensor_data =
sensor_val2,time = ok_date)

            data['result']=sensor_data
    else:
        data['result']='Not Ajax'
    return JsonResponse(data)
```

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
  {% load static %}
<link rel="stylesheet" href="{% static 'bootstrap.min.css' %}">
<link rel="stylesheet" href="{% static 'font-awesome.css' %}">
<script src="{% static 'bootstrap.min.js' %}"></script>
    <style>
        body {
            background-color:darkgray;
        }
        .inputDiv {
            display:flex;
            align-items:center;
            justify-content:center;
```

```
            flex-direction:column;
            height:100%;
            width:100%;
        }
        #button1 {
            margin-left:100px;
            margin-top:50px;
            display:flex;
            align-items:center;
            justify-content:center;



            margin:auto;
        }
        #btn_show {
            padding: 10px 15px;
            width: 250px;
            background-color:black;
        }
        .col-md-4 {
            margin:20px 0px;

        }
       #title{
           margin-left:450px;
           font-size:25px;

        }
    </style>
</head>
<body>
<div class="container">
<center><h1>Live Graph Demo</h1></center>
<hr>

    <div class="row">
        <div class="col-md-12">
            <div class="col-md-4" id="title"><label>Enter COM
Port</label></div>
            <div class="inputDiv">
                <div class="col-md-4">
                    <input type="text" class="form-control col-
md-2" name="com" id="comport">
                </div>
                <div class="col-md-4" id="button1">
                    <input type="submit" name="submit"
id="btn_show" class="btn btn-primary">
                </div>
            </div>
```

```
        </div>
    </div>

<br><br>
<hr/>
<div class="row">
    <div id="chartContainer" style="height: 370px; width:
100%;"></div>
</div>

</div>
</body>

<script src="{% static 'jquery-3.3.1.min.js' %}"></script>
<script
src="https://canvasjs.com/assets/script/canvasjs.min.js"></scrip
t>


<script>
$("#btn_show").click(function(){
    com=$("#comport").val();
    fill_graph_sensor(com);
});


function fill_graph_sensor(){
        var dps = [];
var chart = new CanvasJS.Chart("chartContainer", {
     animationEnabled: true,
     title: {
          text: "Live Data from ESP32 Connected to Load Cell
Sensor"
     },
     axisX: {
          title: "Time"
     },
     axisY: {
          title: "Readings",
          suffix: ""
     },
     data: [{
          type: "line",
          name: "CPU Utilization",
          connectNullData: true,
          xValueType: "dateTime",
          xValueFormatString: "DD MMM hh:mm TT",
          yValueFormatString: "#,#0.00###\"N\"",
          dataPoints: dps
```

```
     }]
});

var xVal = 0;
var yVal = 100;
var updateInterval = 4000;
var dataLength = 20;

var updateChart = function () {

    $.ajax({
            type:"get",
            url: "{% url 'Student:fetch_sensor_values_ajax' %}",
            data: {
                     'id': com
                   },
            success: function(data) {
                console.log(data.result);
                for(index = 0; index < data.result.length;
index++) {
                    var str_array =
data.result[index].split(',');
                    dps.push({
                            x: Date.parse(str_array[1]),
                            y: Number(str_array[0])
                      });
                   }
                return data;
                    },
            error: function(){
                    console.log("error Found!");
                 }
            });

     if (dps.length > dataLength) {
          dps.shift();
     }

     chart.render();
};

updateChart(dataLength);
setInterval(function(){updateChart()}, updateInterval);
}
</script>
</html>
```

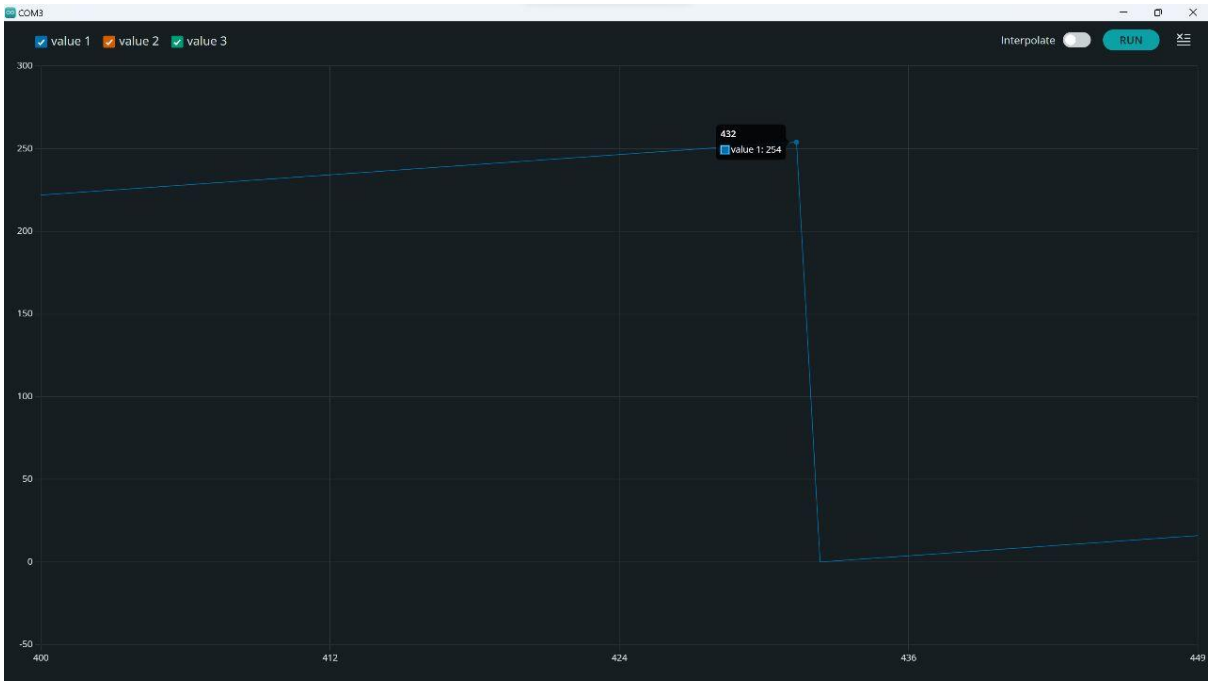
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Live Graph Demo</title>
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/boo
tstrap.min.css" integrity="sha384-
GJzZqFGwb1QTTN6wy59ffF1BuGJpLSa9DkKMp0DgiMDm4iYMj70gZWKYbI706tWS
" crossorigin="anonymous">
        <script src="https://code.jquery.com/jquery-
3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo
" crossorigin="anonymous"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd
/popper.min.js" integrity="sha384-
wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/yoi7FRNXMRBu5WHdZYu1hA6ZOblgut
" crossorigin="anonymous"></script>
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/boots
trap.min.js" integrity="sha384-
B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k
" crossorigin="anonymous"></script>
        {% load staticfiles %}
        <script src="{% static 'jquery-3.3.1.min.js'
%}"></script>
        <script
src="https://canvasjs.com/assets/script/canvasjs.min.js"></scrip
t>

</head>
<body>
    <center>
    <label>Enter COM PORT:</label>
    <input type="text" class="form-control col-md-2" name="com"
id="comport">
    <input type="submit" name="submit" id="btn_show" class="btn
btn-primary">
    <h1>Live data from Ultrasonic Sensor</h1>
        </center>

    <div id="chartContainer" style="height: 450px;
width:100%;"></div>
</body>
<script>
$("#btn_show").click(function(){
    com=$("#comport").val();
    fill_graph_sensor(com);
```

```
});
function fill_graph_sensor(){

    var dps = []; // dataPoints
var chart = new CanvasJS.Chart("chartContainer", {
     animationEnabled: true,
     title: {
          text: "Live Data"
     },
     axisX: {
          title: "Time"
     },
     axisY: {
          title: "data",
          suffix: ""
     },
     data: [{
          type: "line",
          name: "CPU Utilization",
          connectNullData: true,
          xValueType: "dateTime",
          xValueFormatString: "DD MMM hh:mm TT",
          yValueFormatString: "#,#0.00###\"N\"",
          dataPoints: dps
     }]
});

var xVal = 0;
var yVal = 100;
var updateInterval = 4000;
var dataLength = 20;

var updateChart = function () {

    $.ajax({
            type:"get",
            url: "{% url 'Student:fetch_sensor_values_ajax' %}",
            data: {
                     'id': com
                   },
            success: function(data) {
                console.log(data.result);
                for(index = 0; index < data.result.length;
index++) {
                    var str_array =
data.result[index].split(',');
                    dps.push({
                           x: Date.parse(str_array[1]),
                           y: Number(str_array[0])
```

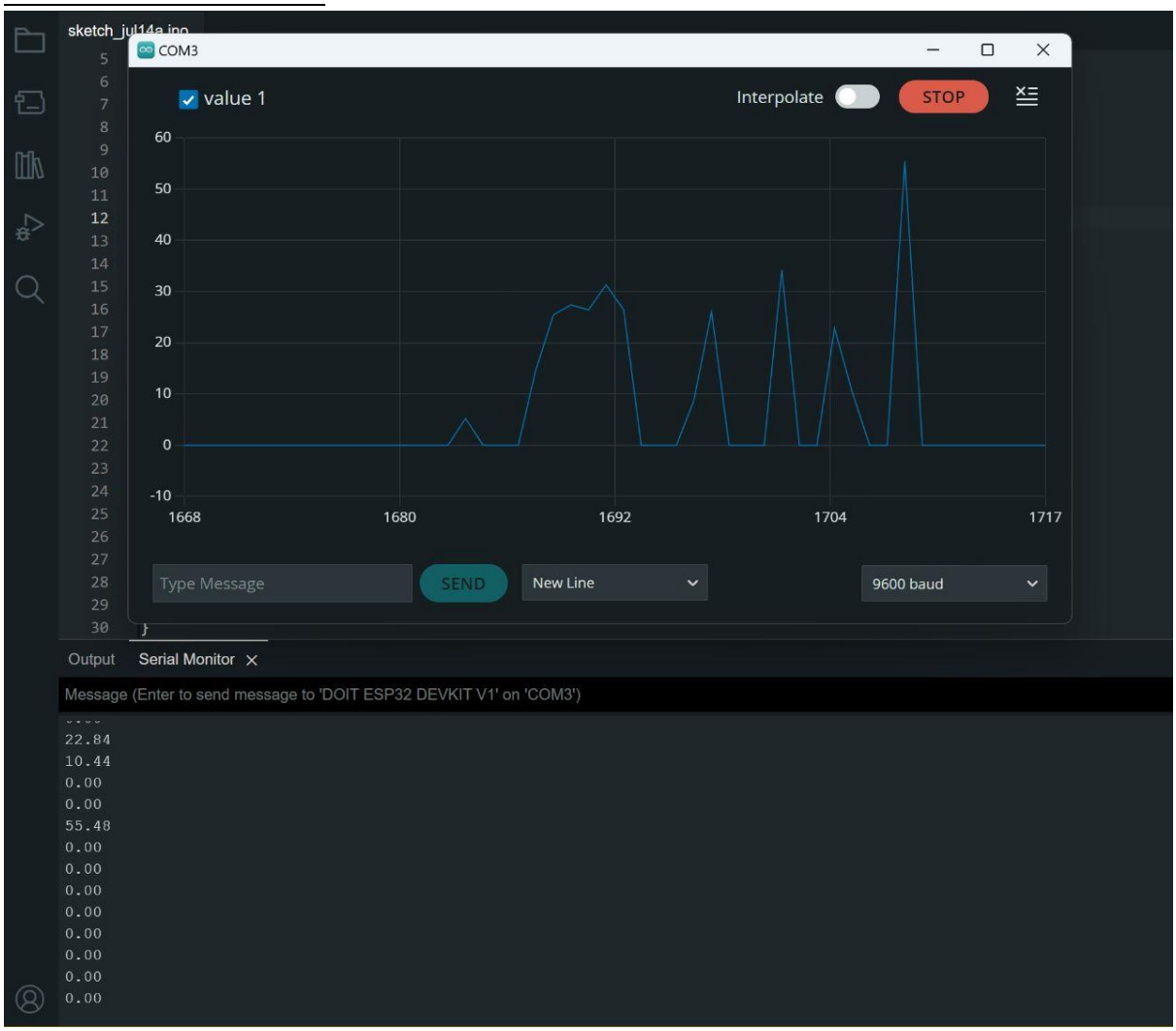
```
                    });
                }
            return data;
                },
        error: function(){
                console.log("error found!");
            }
        });

    if (dps.length > dataLength) {
        dps.shift();
    }

    chart.render();
};

updateChart(dataLength);
setInterval(function(){updateChart()}, updateInterval);


}
</script>
</html>
```

Results:

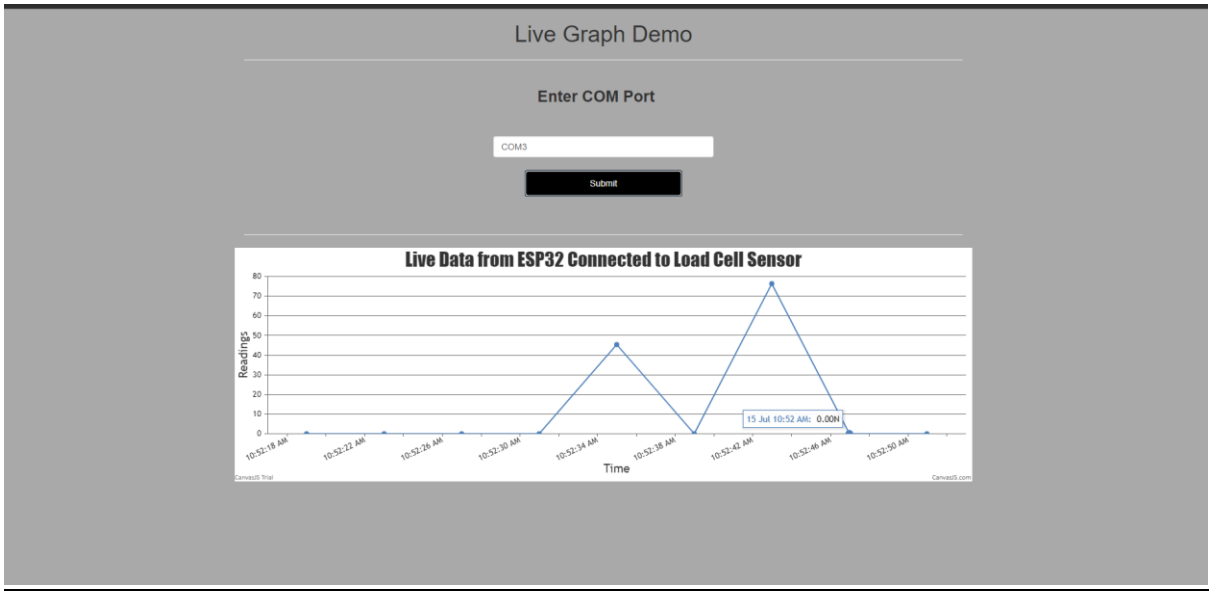
Serial input from ESP 32 to create a Sinewave



Arduino Serial Plotter



Django Implementation:



Conclusion:

In conclusion, the utilization of the SK6511 load cell sensor in conjunction with the JY-S60 weight transmitter provides an effective and accurate force measurement solution. The load cell sensor, known for its high precision and reliability, serves as the primary component for measuring the applied force. Its design ensures minimal interference and consistent performance across various applications.

The JY-S60 weight transmitter acts as a crucial intermediary between the load cell sensor and the measurement system. It receives the analog signals from the load cell and converts them into digital data, making it compatible with a wide range of measurement and control devices. Additionally, the transmitter offers features such as signal amplification, filtering, and calibration options, enabling enhanced accuracy and flexibility in force measurement.

When combined, the SK6511 load cell sensor and JY-S60 weight transmitter create a robust force measurement system suitable for diverse industrial, research, and engineering applications. Whether it is used for material testing, force monitoring, or quality control, this integrated solution delivers reliable and precise force measurements.

Furthermore, the ease of installation and setup of both the load cell sensor and weight transmitter contribute to the overall convenience and efficiency of the force measurement process. With proper calibration and regular maintenance, this system can provide accurate force measurements over an extended period.

In summary, the SK6511 load cell sensor and JY-S60 weight transmitter offer a powerful force measurement solution characterized by accuracy, reliability, and versatility. Their combined capabilities make them an excellent choice for a wide range of force measurement applications, contributing to improved efficiency, quality control, and data-driven decision-making.

References:

[1] Sonsare, Pravin. (2018). IOT based Smart weighing system for Crate in Agriculture. International Journal of Computer Sciences and Engineering. 6. 336-341. 10.26438/ijcse/v6i1.336341.

[2] Riyanti, Kurnia & Kakaravada, Ismail & Ali Ahmed, Abdussalam. (2022). Automatic Load Detector Design to Determine the Strength of Pedestrian Bridges Using Load Cell Sensor Based on Arduino. Indonesian Journal of Electronics Electromedical Engineering and Medical Informatics. 4. 16-22. 10.35882/ijeeemi.v4i1.3.

[3] Karar, Mohamed & Shehata, Hazem & Reyad, Omar. (2022). A Survey of IoT-Based Fall Detection for Aiding Elderly Care: Sensors, Methods, Challenges and Future Trends. Applied Sciences. 12. 3276. 10.3390/app12073276.

[4] Dishari Sengupta, Subhadip Ray, Progati Biswas, Subham Rajak, Dr. Debasish Mondal (2021). Smart Fall Detection System for Elderly People with IOT and Sensors.

[5] Müller, Ivan & Machado de Brito, Renato & Pereira, Carlos & Brusamarello, Valner. (2010). Load cells in force sensing analysis - Theory and a novel application. Instrumentation & Measurement Magazine, IEEE. 13. 15 - 19. 10.1109/MIM.2010.5399212.