Bridge Safety System using IoT

A smart and innovative system that ensures the safety of bridge users.

Explore the technology behind this system.

By Tushar Paul

Email ID : tusharpaul2001@gmail.com

The Objective

1 Promote Safety

The primary objective of this system is to promote safety for every person that uses the bridge. It's essential to monitor the weight distribution and detect any vibrations that may cause accidents.

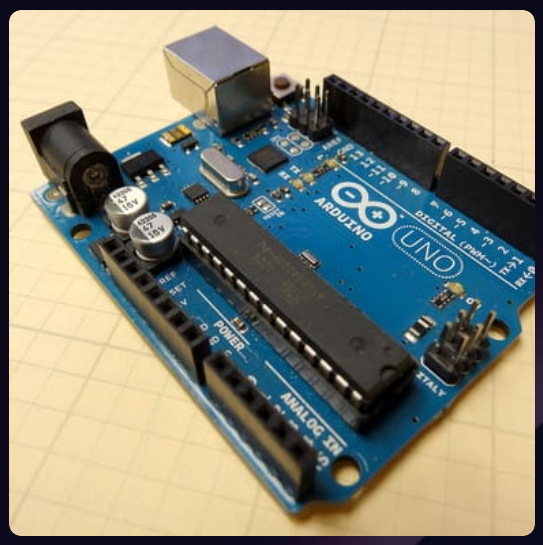
2 Real-Time Monitoring

As this system is an IoT-based technology, it offers a real-time monitoring facility by capturing essential metrics like load, vibration, and temperature. This will notice any possible anomalies and allow for immediate intervention.

3 Early Warning System

The objective of this system is to have an early warning system that signals any fault that may happen so that action is taken before it's too late.

The Components

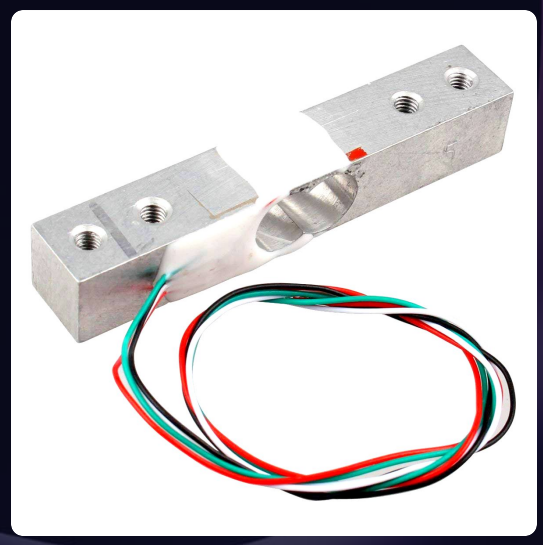


Arduino

The Arduino is the brain behind this system that connects all the essential components and communicates with the server. It takes the inputs from sensors and sends them to the cloud server for analysis.

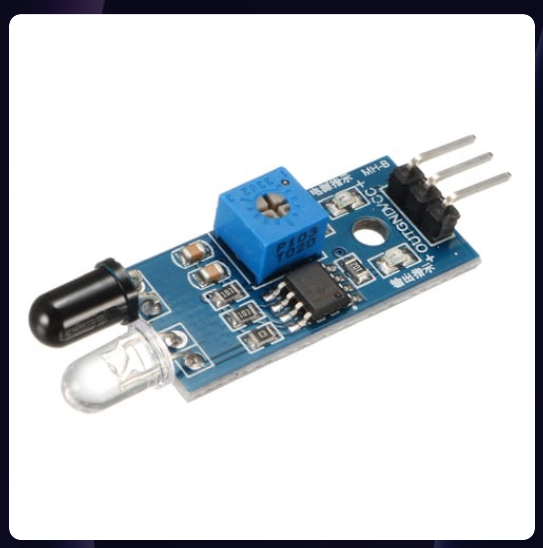
ESP32

The ESP32 is a low-powered device that connects the Arduino to the internet. It's responsible for receiving data from the Arduino and sending it to the cloud server via Wi-Fi connection.



Load Sensor

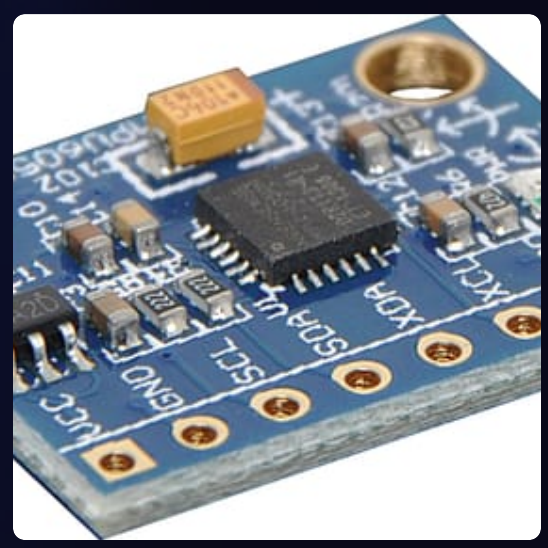
The Load Sensor detects the weight distribution of the vehicle that passes through the bridge and sends it to the Arduino for analysis and interpretation.



IR Sensor

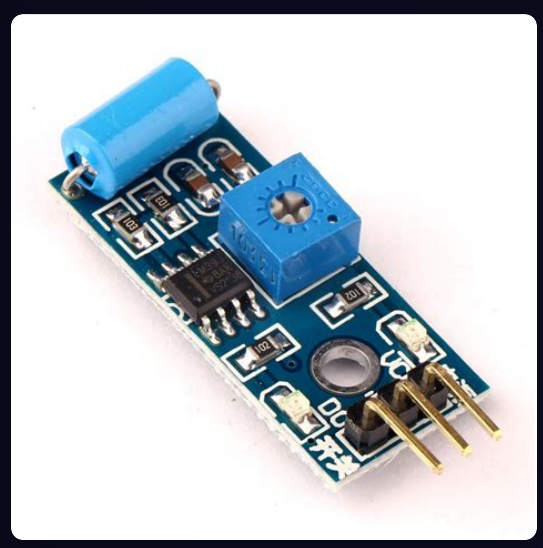
The IR Sensor is responsible for detecting the vibrations and providing the Arduino with valuable feedback so that appropriate measures can be taken.

.



MPU 650

The MPU 650 measures the angle of the bridge and provides the necessary feedback to the Arduino to ensure the integrity of the bridge geometry.



Vibration Sensor

Measures vibration levels in machinery for screening and analysis

Methodology

Testing

The system is tested to ensure proper connection between the sensors, and the data is recorded to detect if there are any inconsistencies.

1

Assembly

The components are assembled and connected via cables, and necessary settings are made on the Arduino and ESP32.

2

3

Data Collection

The sensors are activated and start to collect data about the bridge, which is then sent to the cloud server for analysis.

The Working of the System

Data Collection

The system starts collecting data as the vehicle passes through the bridge, the weight on the bridge is measured by the Load Sensor, vibrations by IR Sensor & angles by MPU 650. The data is sent to the Arduino.

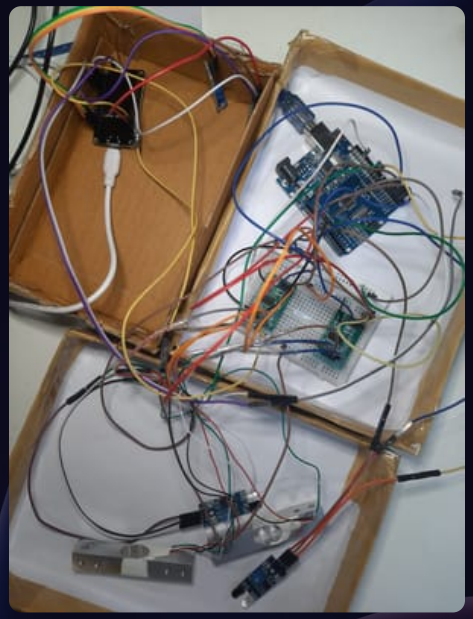
Data Processing

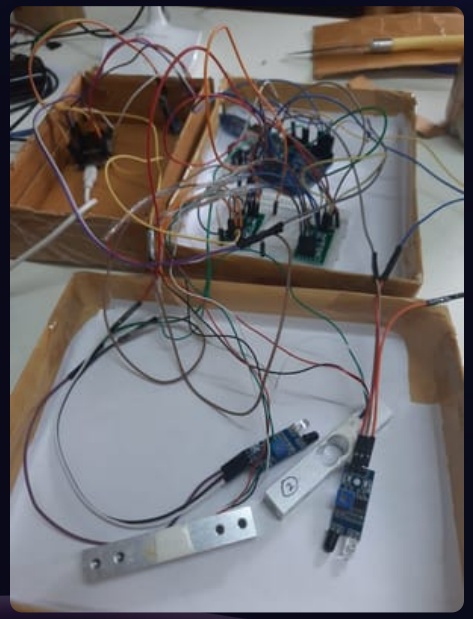
The Arduino receives the data and preprocesses it by performing necessary calculations. The data is then sorted, analyzed, and checked for accuracy.

Action

The analyzed data is sent to the cloud server, which then sends it to necessary stations, triggering alarms if necessary, and analysis and decision-making.

SETUP





sketch_may18a | Arduino 1.8.19

File Edit Sketch Tools Help

sketch_may18a

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>


const int vibrationSensorPin = 15;

Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(115200);

  pinMode(vibrationSensorPin, INPUT); // Set vibration sensor pin as input

  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");
```

Done uploading.

```
Writing at 0x000465ae... (80 %)
Writing at 0x0004c2b4... (90 %)
Writing at 0x00051848... (100 %)
Wrote 285456 bytes (159474 compressed) at 0x00010000 in 14.3 seconds (effective 159.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

ESP32-WROOM-DA Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 115200, Core 1, Core 1, None, Disabled on COM5

Esp32_Load_SNesor | Arduino 1.8.19

File Edit Sketch Tools Help

Esp32_Load_SNesor

```
    v =0;
    Serial.println("No vibration detected.");
  }

  delay(1000); // Delay for 1 second before the next readin
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  /* Print out the values */
  Serial.print("Acceleration X: ");
  Serial.print(a.acceleration.x);
  float a1 = a.acceleration.x;
  Serial.print(", Y: ");
  Serial.print(a.acceleration.y);
  float a2 = a.acceleration.y;
  Serial.print(", Z: ");
  Serial.print(a.acceleration.z);
  float a3 = a.acceleration.z;
  Serial.println(" m/s^2");

  Serial.print("Rotation X: ");
  Serial.print(g.gyro.x);
  float g1 = g.gyro.x;
  Serial.print(", Y: ");
  Serial.print(g.gyro.y);
  float g2 = g.gyro.y;
  Serial.print(", Z: ");
  Serial.print(g.gyro.z);
```

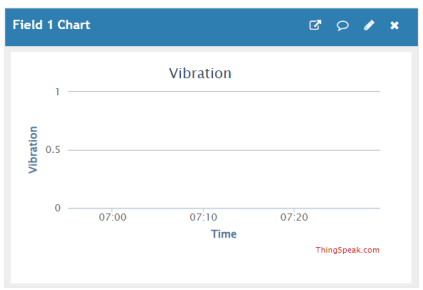
Done uploading.

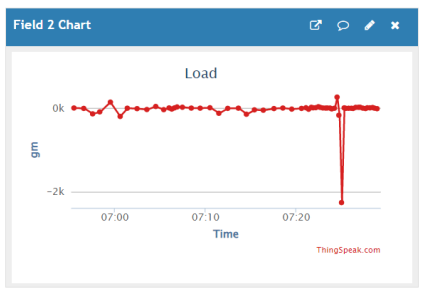
```
Writing at 0x000e4813... (97 %)
Writing at 0x000e9c35... (100 %)
Wrote 912272 bytes (587485 compressed) at 0x00010000 in 52.6 seconds (effective 138.9 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

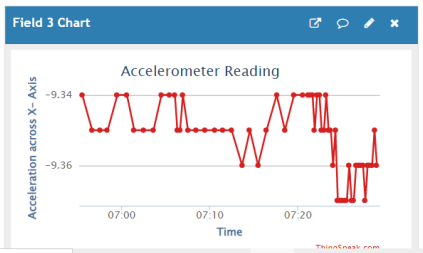
ESP32-WROOM-DA Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 115200, Core 1, Core 1, None, Disabled on COM5

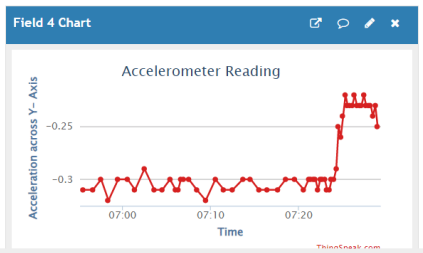
ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy TP

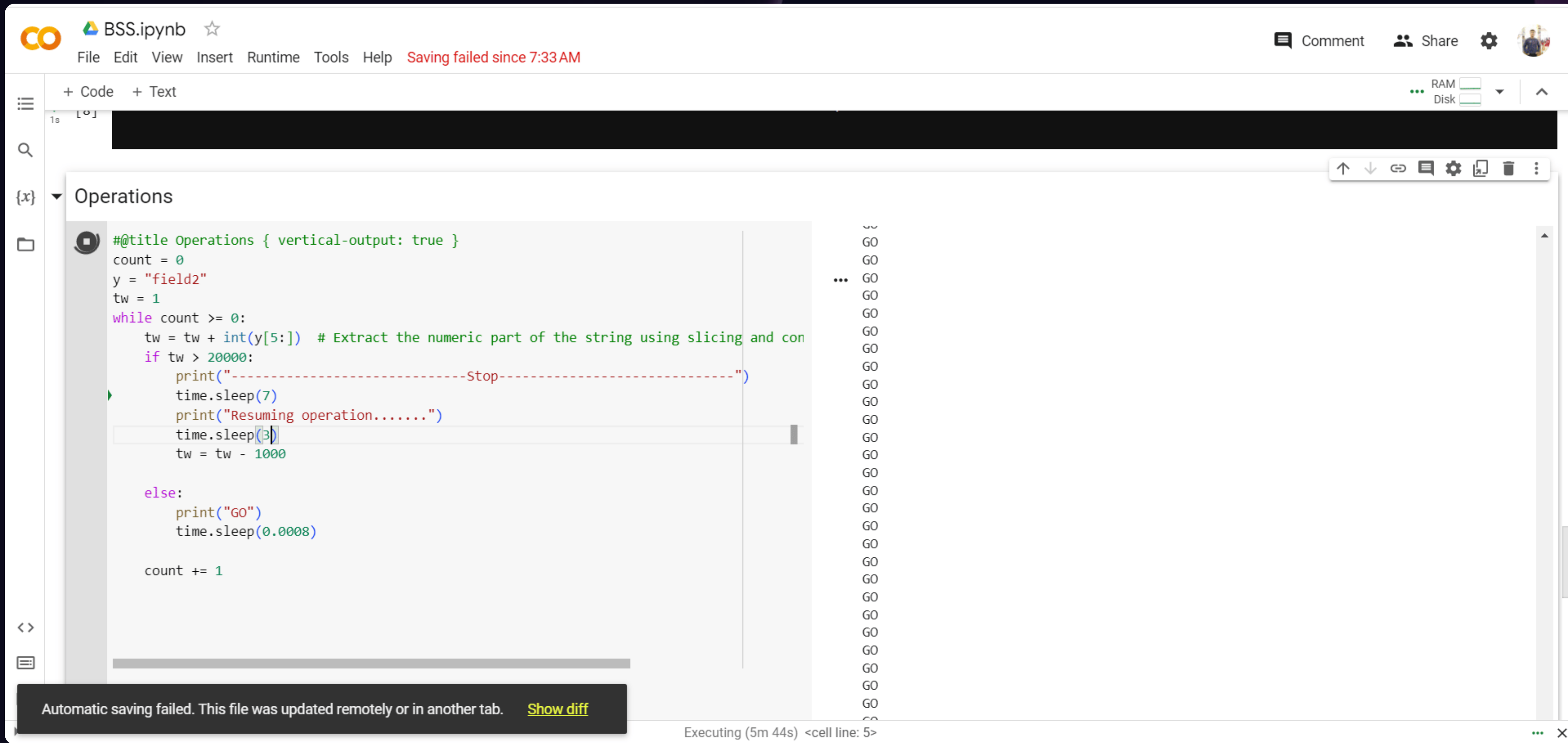
Entries: 1698

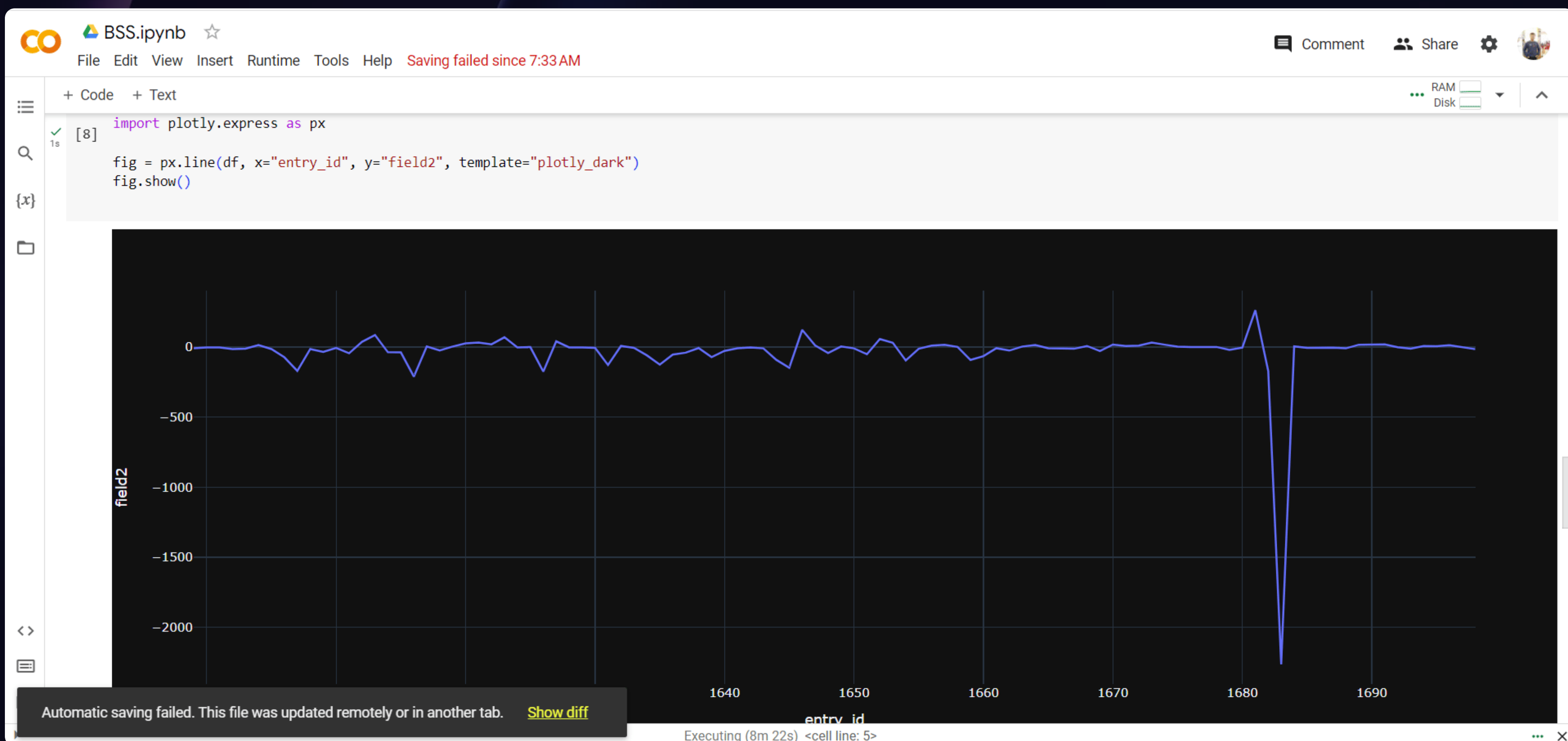









BSS.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 7:33 AM
Comment
Share
+ Code + Text
RAM
Disk
Operations
#@title Operations { vertical-output: true }
count = 0
y = "field2"
tw = 1
while count >= 0:
tw = tw + int(y[5:]) # Extract the numeric part of the string using slicing and con
if tw > 20000:
print("---------------------------Stop---------------------------")
time.sleep(7)
print("Resuming operation.......")
time.sleep(3)
tw = tw - 1000
else:
print("GO")
time.sleep(0.0008)
count += 1
GO
Automatic saving failed. This file was updated remotely or in another tab. Show diff
Executing (5m 44s) <cell line: 5>


BSS.ipynb
File Edit View Insert Runtime Tools Help Saving failed since 7:33 AM
Comment
Share
+ Code + Text
RAM
Disk
[8]
import plotly.express as px
fig = px.line(df, x="entry_id", y="field2", template="plotly_dark")
fig.show()
0
−500
−1000
−1500
−2000
field2
1640
1650
1660
1670
1680
1690
entry_id
Automatic saving failed. This file was updated remotely or in another tab. Show diff
Executing (8m 22s) <cell line: 5>

Visualizing and Calculating Results on ThingSpeak and Colab

ThingSpeak

The system sends the data to ThingSpeak, which prepares it in the form of reports, graphs, and other visualizations, making it easy to identify trends and patterns.

Colab

The data is pre-processed on Colab, which offers and calculates results giving a better understanding of the actual weight distribution on the bridge.

The Benefits of the Bridge safety System

1

Advanced Safety

The Bridge Safety System ensures that the drivers, pedestrians, commuters, and other people on the bridge are safe and that there is no overloading, which can cause structural damage.

2

Timely Intervention

By sending data to the server, the system allows for timely intervention to prevent any accidents and protect people on the bridge.

3

Cost-Effective

The system helps in early detection of any anomalies on the bridge making any necessary maintenance cheaper, thus reducing costs substantially.

Conclusion

The Bridge Safety System using IoT follows a systematic approach that ensures the safety of people using the bridge. By providing real-time monitoring and early detection of anomalies, the system guarantees accidents are prevented, saving lives, and reducing costs while improving overall reliability.

Thank You