DIGITAL TWIN MONITERING SYSTEM

By

Hanchi Ge

Kowshik Dey

Rongjian Chen

Final Report draft for ECE 445, Senior Design, Spring 2024

TA: Xinlong Huang

10 May 2024

Project No. 43

Abstract

Digital twin is a technology that connects the physical and digital realms. By creating a digital copy of the physical system, the technology can monitor, analyze, and predict the performance of the system in different scenarios in real time. The digital twin is constantly updated with data from sensors and other relevant sources, ensuring that it accurately reflects the current state of its physical counterpart. This paper describes our work on creating a digital twin analog bridge equipped with strain gauges to measure the weight of passing vehicles, which also incorporates real-time data and a traffic light-based alert system. This study elucidates new design methods for building physical models, as well as digital twins, and then integrating machine learning methods into the hardware.

Contents

1. Introduction

The focus of our project, the Digital Twin Monitoring system, is to develop a dynamic solution that bridges the gap between the physical and digital model through a real-time, data-driven replica of the physical model bridge. This system is designed to enhance our ability to monitor, analyze and predict the structural integrity and traffic efficiency on the bridge using a suite of integrated measurements and machine learning algorithms.

The primary challenge of our project is the evident need for a more sophisticated, real-time monitoring system that can predict and alert for structural and traffic-related concerns on the bridge. This is especially critical for the prevention of accidents in the highway and ensuring timely maintenance resulting in saving thousands of dollars. Our proposed solution involves the use of digital twin simulation coupled with a traffic light alert system based on the weight of the crossing vehicles.

The digital twin monitoring system is comprised of multiple subsystems, including the strain gauge module for mechanical stress measurement, a data processing controller, and a communication system for real-time data transmission and notifications. These subsystems work in conjunction to facilitate the ongoing monitoring and evaluation of the bridge's structural integrity and traffic patterns. Here we provide a brief overview of how each of the component contributes to the overarching goal of the project and how they are interconnected as follows:

- **Strain Gauge Module:** This component measures the mechanical strain on the bridge, providing data that is critical for accessing structural integrity. It is connected to the Wheatstone bridge circuit for accurate measurement conversion.
- **Controller:** The controller module comprises of a STM32 processor that uses the data from the strain gauge at critical points and other sensors, using a machine learning(SVC) model to predict the structural failures and to manage the traffic light system.
- **Traffic Light System:** Receives control signals from the controller to manage traffic flow based on the bridge's current load capacity.
- **Communication System:** This module ensures real-time data transfer between the bridge and the monitoring station/phones via Bluetooth module, facilitating immediate responses to detected issues.
- **Voltage Regulator:** Controls the voltage input to the strain gauge and all the other modules.

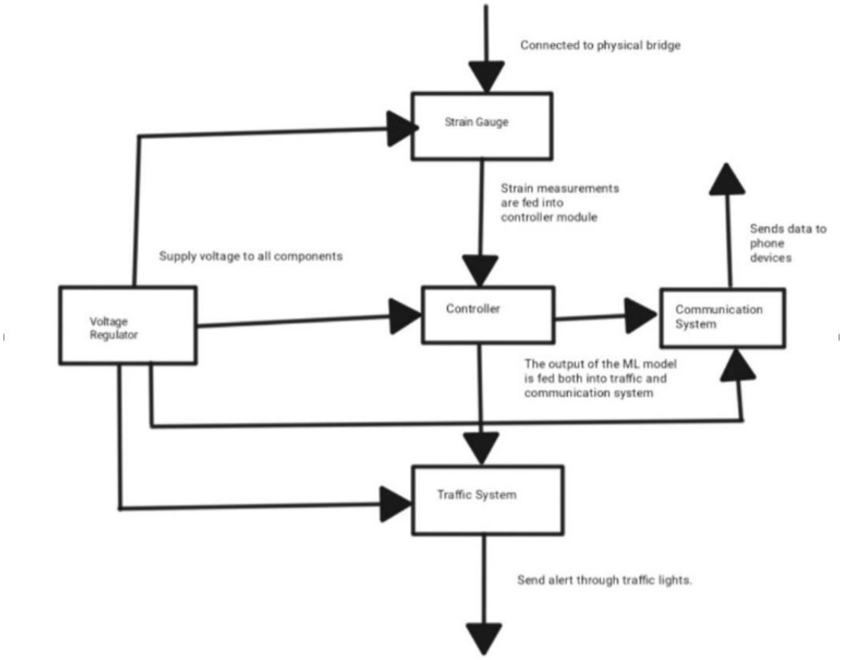


Figure 1. Top level diagram

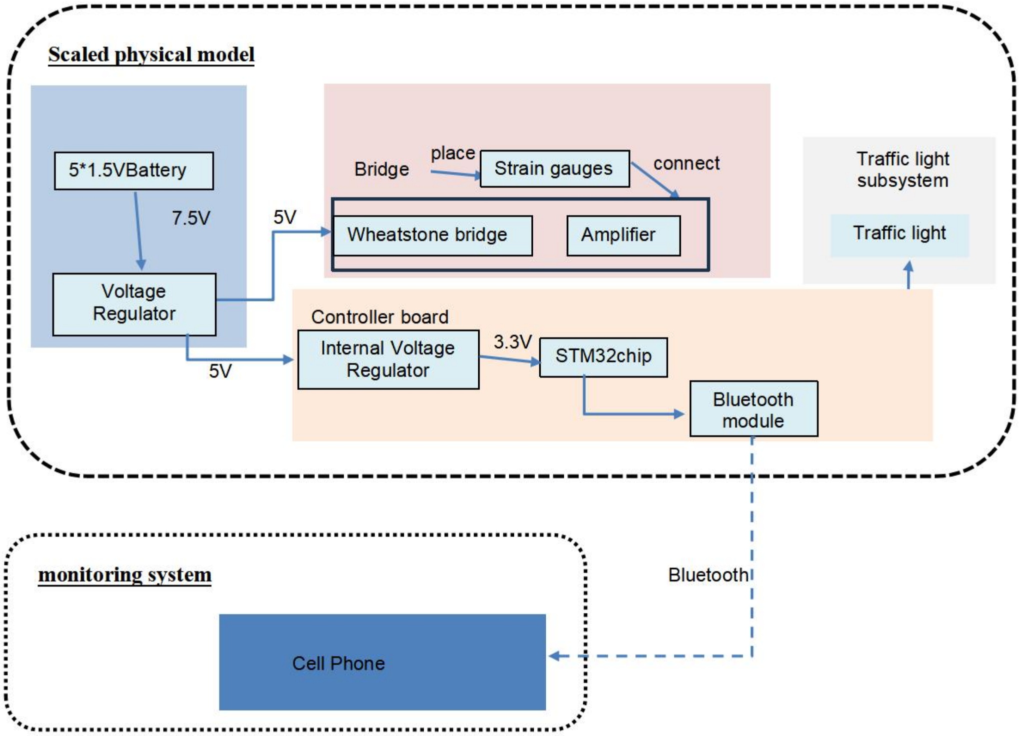


Figure 2. Block diagram

2 Design

Our system is a combination of subsystems. A key part of our system is the strain gauge module, which is designed to detect and measure mechanical strain with a precisely calibrated Wyeth bridge and amplified output to improve signal clarity. This input is fed into a control module powered by a dedicated voltage regulator to ensure the stable operation of the subsystem. The control module uses STM32F103C8T6 microcontroller to manage the data of strain gauge and execute the control command. Wireless communication via Bluetooth interface allows real-time data transfer to monitor data. The system's functionality uses machine learning to predict bridge structural safety. Below, we will delve into the details of each module, highlighting their individual contributions to the overall functionality of the system.

2.1 Strain gauge module

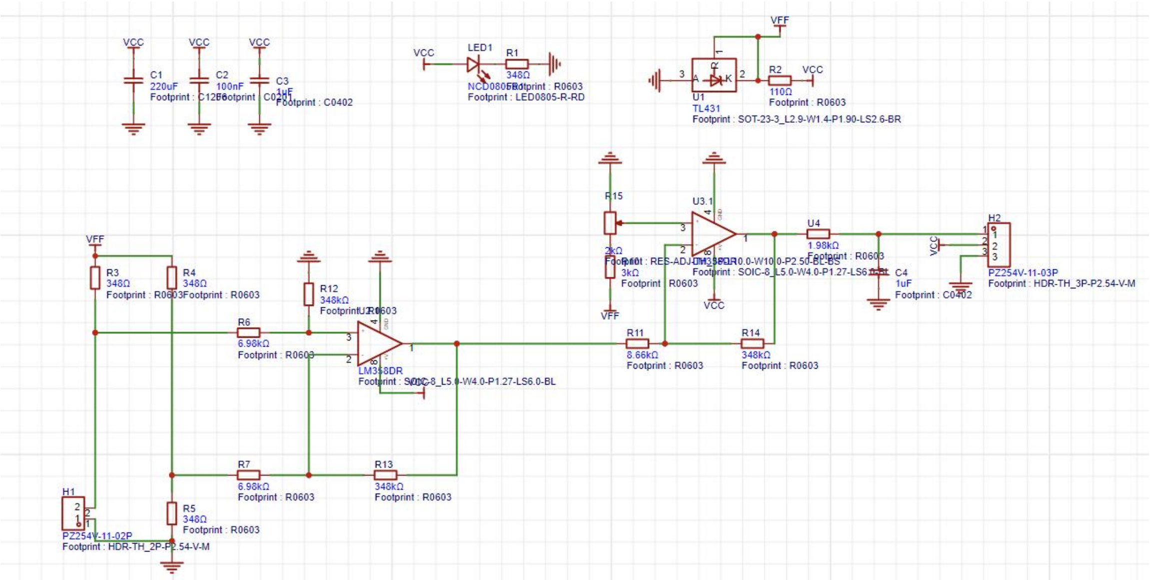


Figure3. Sch of strain gauge module

2.1.1 Strain gauge

A strain gauge is a sensor whose resistance varies with applied force. It converts force, pressure, tension, weight, etc., into a change in electrical resistance which can then be measured. In your schematic, H1 connects to an external strain gauge. When attached to an object, the strain gauge will deform when the object is subjected to stress, and this deformation will change the electrical resistance of the gauge.

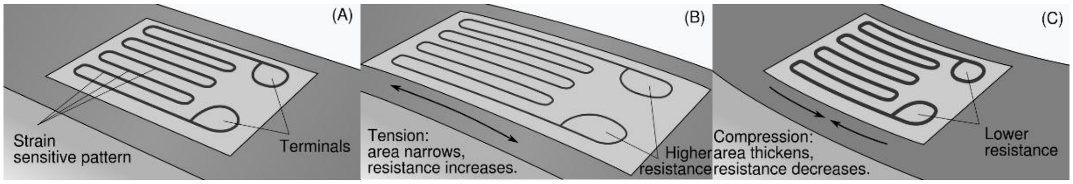


Figure4. Resistance of strain gauge

2.1.2 Wheatstone bridge

$$e_0 = \frac{R_3 R_5 - R_s R_4}{(R_3 + R_s)(R_5 + R_4)} \times VFF$$

A Wheatstone bridge is a circuit used to accurately measure resistance. In the application of strain gauges, Bridges help detect small changes in resistance. The voltage on the bridge changes as the resistance of the strain gauge changes, which occurs when strain is applied to the gauge.

2.1.3 Amplifier

The output of a Wheatstone bridge is a small voltage change due to strain that is usually insufficient for direct processing or reading. Therefore, we use amplifiers to increase the amplitude of this voltage to a more detectable level. In my schematic, two LM324DR amplifiers are used. The IC amplifies the voltage variation of the Wheatstone bridge, making it easier to measure and interpret. The output interface of the module is H2, and pin 1 is the output voltage.

2.2 Voltage regulator

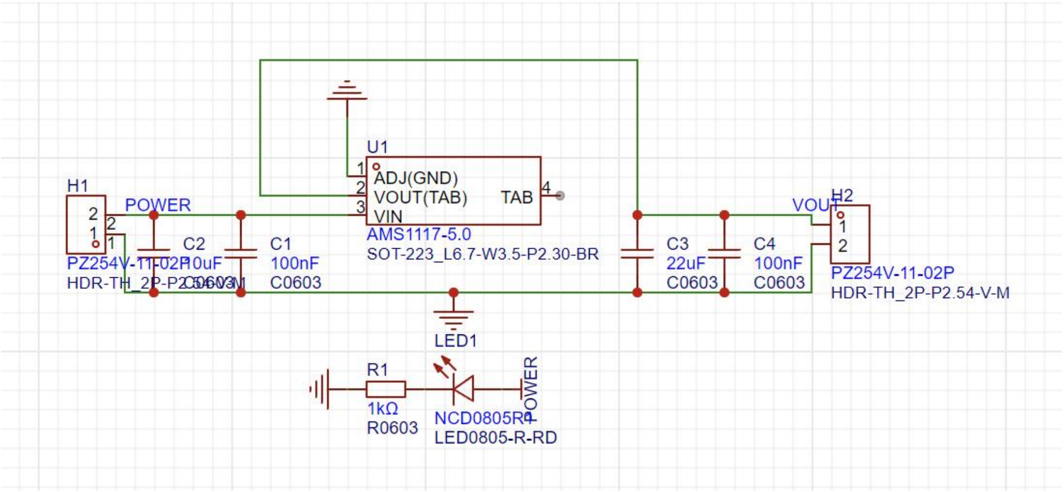


Figure5. Sch of Voltage regulator

Since the strain gauge module requires 5V input voltage, we made this regulator module to supply 5V voltage to the other subsystems.

The module described in the schematic is centered on the AMS1117-5.0, which is a low difference (LDO) regulator. The regulator is designed to accept input voltage, supplied via an H1 connector, from a power supply consisting of five dry cells connected in series. These batteries typically produce a voltage of around 7.5V. The AMS1117-5.0 efficiently converts this input into a stable 5V output, which is critical for powering sensitive electronic devices that require consistent voltage levels. The module's design includes additional components such as decoupling capacitors to stabilize input and output voltages, and LED indicators with series resistors to signal when the power supply is correct and the regulator is working. This configuration makes the module ideal for a variety of applications that require a reliable and constant 5V power supply.

2.3 Controller

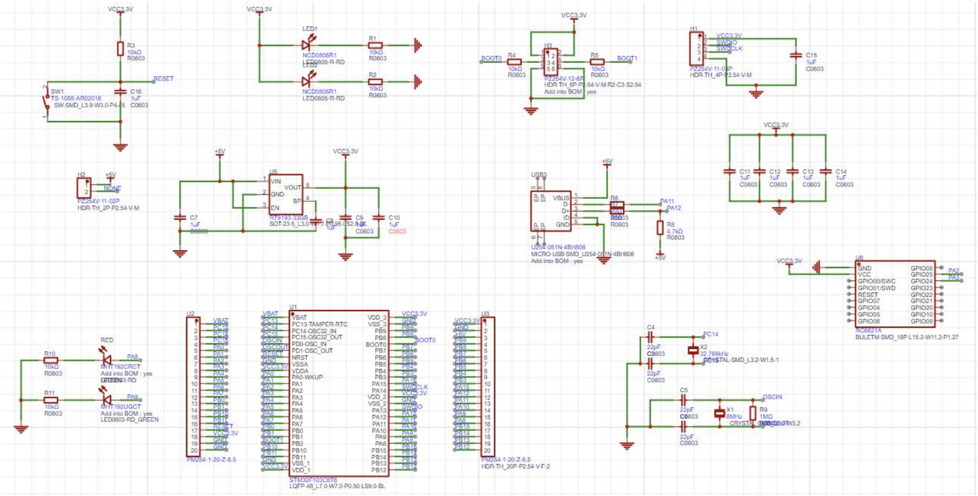


Figure6. Sch of Controller

2.3.1 stm32f103c8t6

The STM32F103C8T6 is a high-performance microcontroller of the STM32 family with a 32-bit ARM Cortex-M3 processor. The microcontroller is ideal for embedded applications that require a good balance between performance and cost. It includes a variety of features such as multiple GPIOs (universal input/output), advanced peripherals such as ADCs (analog-to-digital converters), USARTs, SPI and I2C interfaces. The microcontroller acts as the central processing unit in the controller module, managing the input from the sensor and controlling the output. Through the SWD interface, we can use the STM32cubeIDE to burn the microcontroller, so as to achieve the functions we want, such as receiving Bluetooth signals and controlling the status of the traffic light.

2.3.2 Bluetooth RC6621A

The RC6621A component in the schematic refers to a Bluetooth module capable of providing wireless connectivity. The module enables the controller to communicate with other Bluetooth-enabled devices such as smartphones, computers or other controllers, facilitating data transfer and remote control functions. This Bluetooth module is essential for wireless communication applications that need to send or receive commands and data without a physical connection. Through this Bluetooth module, we can transmit the strain gauge module signal and signal level status received by the control module to the mobile phone or computer in real time.

2.3.3 5v-3.3v voltage regulator

The voltage regulator circuit is designed to convert a 5V power supply to 3.3V, which is required for many components on the board, including the STM32F103C8T6 and Bluetooth modules. The regulator ensures that microcontrollers and other sensitive electronic components receive a stable power supply, providing the power necessary for reliable operation. The use of such regulators is essential to match the voltage requirements of various components to ensure module life and reliability.

2.3.4 Crystal oscillator circuit

The crystal oscillator circuit provides the precise clock signal for the microcontroller. It consists of two quartz crystals and four capacitors that help stabilize the oscillation frequency. This clock is critical for a microcontroller because it determines the speed at which it operates and ensures timing accuracy for tasks such as communication protocols, sensor readings, and time-related calculations. The correct and stable operation of the crystal oscillator is the basis of the reliable performance of the whole controller system.

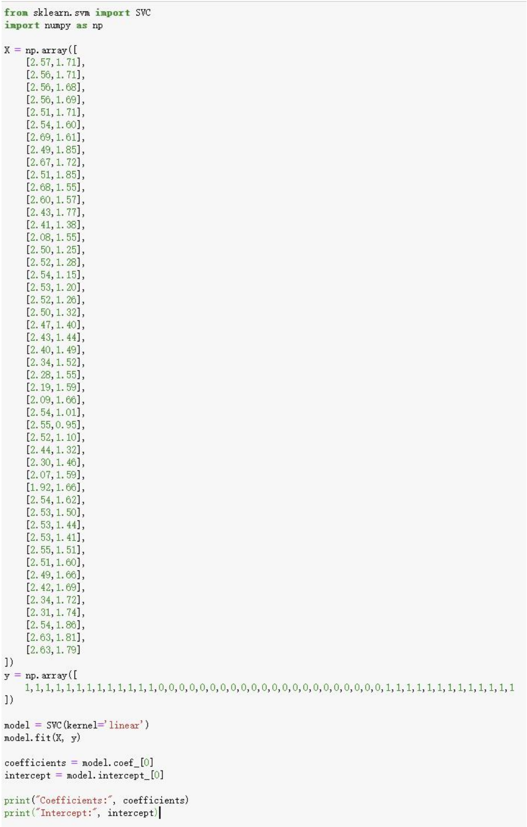
2.3.5 Traffic light

In the circuit, the PA8 and PA9 pins of the stm32f103c8t6 control the LED red and LED green respectively. The light goes green when our program determines that the bridge structure is safe. When a heavy vehicle passes by, causing the bridge to deform too much, the red light goes on and stays on for five seconds.

2.4 Code

The code was an integral part of our project, and we first used a support vector machine on python to learn the signal of the strain gauge module in different situations. It is then simplified to a pattern of two coefficients and an intercept. For the code burned to the chip, we use C to complete the writing of main.c, so that the control subsystem can realize traffic light control and send signals through Bluetooth.

2.4.1 Support vector machine



```python
from sklearn.svm import SVC
import numpy as np

X = np.array([
    [2.57,1.71],
    [2.56,1.71],
    [2.56,1.68],
    [2.56,1.69],
    [2.51,1.71],
    [2.54,1.60],
    [2.69,1.61],
    [2.49,1.85],
    [2.67,1.72],
    [2.51,1.85],
    [2.68,1.55],
    [2.60,1.57],
    [2.43,1.77],
    [2.41,1.38],
    [2.08,1.55],
    [2.50,1.25],
    [2.52,1.28],
    [2.54,1.15],
    [2.53,1.20],
    [2.52,1.26],
    [2.50,1.32],
    [2.47,1.40],
    [2.43,1.44],
    [2.40,1.49],
    [2.34,1.52],
    [2.28,1.55],
    [2.19,1.59],
    [2.09,1.66],
    [2.54,1.01],
    [2.55,0.95],
    [2.52,1.10],
    [2.44,1.32],
    [2.30,1.46],
    [2.07,1.59],
    [1.92,1.66],
    [2.54,1.62],
    [2.53,1.50],
    [2.53,1.44],
    [2.53,1.41],
    [2.55,1.51],
    [2.51,1.60],
    [2.49,1.66],
    [2.42,1.69],
    [2.34,1.72],
    [2.31,1.74],
    [2.54,1.86],
    [2.63,1.81],
    [2.63,1.79]
])
y = np.array([
    1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1
])

model = SVC(kernel='linear')
model.fit(X, y)

coefficients = model.coef_[0]
intercept = model.intercept_[0]

print("Coefficients:", coefficients)
print("Intercept:", intercept)
```

Figure7. Code of SVM

2.4.2 main.c of stm32

```
#include "main.h"
#include "adc.h"
#include "usart.h"
#include "gpio.h"

float decision_score;
void SystemClock_Config(void);
uint32_t readADC(uint32_t channel)
{
  ADC_ChannelConfTypeDef sConfig = {0};
  sConfig.Channel = channel;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  HAL_ADC_ConfigChannel(&hadc1, &sConfig);
  HAL_ADC_Start(&hadc1);
  HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
  return HAL_ADC_GetValue(&hadc1);
}
int main(void)
{
 HAL_Init();
 SystemClock_Config();
 MX_GPIO_Init();
 MX_ADC1_Init();
 MX_ADC2_Init();
 MX_USART2_UART_Init();
  uint32_t adcValue1, adcValue2;
  float decision_score;
  char message[100];
  while (1)
  {
    adcValue1 = readADC(ADC_CHANNEL_0); // Replace with the ADC channel that is actually connected to strain gauge 1
    adcValue2 = readADC(ADC_CHANNEL_1); // Replace with the ADC channel that is actually connected to strain gauge 2
    GPIO_PinState outStatePA8 = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8); // Read the status of PA8
    GPIO_PinState outStatePA9 = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_9); // Read the status of PA9

    // formatting
    snprintf(message, sizeof(message), "ADC0: %lu, ADC1: %lu, PA8: %d, PA9: %d\r\n",
         adcValue1, adcValue2, outStatePA8, outStatePA9);
    // Send data via USART
    HAL_UART_Transmit(&huart2, (uint8_t*)message, strlen(message), HAL_MAX_DELAY);
    HAL_Delay(100); // 0.1s
    // Calculate linear decisions using the coefficients and intercepts obtained from the Python model
    decision_score = 2.854 * adcValue1/4096*3.3 + 3.469 * adcValue2/4096*3.3 - 10;
    if (decision_score > 0) {
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);   // Set PA9 high (green)
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET); // Set PA8 low (red off)
    } else {
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_RESET); // Set PA8 low (green off)
```

```
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);   // Set PA9 high (red)
          HAL_Delay(5000);                                 // Hold red for 5 seconds
        }
      }
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV2;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC;
  PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV4;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  __disable_irq();
  while (1)
  {
  }
```

```
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

3. Design Verification

To ensure the reliability of our project design, a validation process is implemented in all modules. This process includes functional testing under laboratory conditions and simulation scenarios. From the strain gauge module to the regulator and controller, each component is tested to ensure it meets operational requirements. These tests are essential to confirm the reliability of our systems. Below, we will detail the specific validation process and results for each subsystem.

3.1 Strain gauge module

To actually verify the strain gauge module, we first built a prototype on the PCB, and after soldering, made sure that all the connections were secure. We try forward and reverse bending strain gauges to provide stress. Under ideal conditions, $Vout = G \times\ (\epsilon \times S) \times e_0$, V is the output voltage, G is the multiplier, and S is the sensitivity. In the actual case, we only need to verify that the output signal of the strain gauge module always becomes larger or smaller when increasing stress is provided in one direction.

In fact, in the test, because we cannot confirm how much stress is, we can only judge the size of the stress by the degree of bending, which is a deficiency in the test.

3.2 Voltage regulator

This is a relatively simple module. In the laboratory, we connected the generator to H1, provided input voltage, varied the input voltage within the range of 6.5-12V, and tested that the output voltage was always 5V, confirming the success of the test.

In this test, we found that when the input voltage was 5-6V, the output voltage did not work properly, so we chose 6.5-12v as the test range.

3.3 Controller

When all PCB are made, all components are soldered and properly connected, we begin verification.

Traffic light: When a remote-controlled car drove over our model bridge, the structure was safe and the green light remained on. Gradually increase the weight loaded on the car until the red light comes on and remains off for five seconds after the car leaves, and the green light continues to come on after that. It's the same thing as the weight goes up.

Bluetooth: The TX of stm32 is connected with the RX of Bluetooth, and the RX of stm32 is connected with the TX of Bluetooth to complete asynchronous communication and realize serial communication. On the Bluetooth app of the mobile phone, the signal of the two strain gauge modules and the LED level signal can be received and shown. The test was successful.

4. Costs

4.1 Parts

Table 1 Parts Costs

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
Aluminum Beam (Taobao)	Shengjili	118.8 yuan	118.8 yuan	118.8 yuan
LED (Taobao)	Zave	2.68 yuan	2.68 yuan	2.68 yuan
Strain Gauge (Taobao, BFH120-5AA)	Guantal Electronics	60 yuan	30 yuan	60 yuan
Data acquisition board	Lexus Electronics	339 yuan	339 yuan	339 yuan
Battery Case	Risym	2.76 yuan	2.76 yuan	2.76 yuan
PCB Board	Jiali Chuang Electronics	46.8 yuan	46.8 yuan	46.8 yuan
Strain Gauge Module	Jiali Chuang Electronics	46 yuan	46 yuan	46 yuan
controller	Jiali Chuang Electronics	46 yuan	46 yuan	46 yuan
Weights	Yanheng	5.3 yuan	5.3 yuan	53 yuan
Total	**/**	751.94 yuan	721.94 yuan	799.64 yuan

4.2 Labor

Labor	Rongjian Chen	87769 dollars / year	130 hours	3908 dollars
	Hanchi Ge	87769 dollars / year	130 hours	3908 dollars
	Kowshik Dey	87769 dollars / year	130 hours	3908 dollars

5. Conclusion

5.1 Accomplishments

The strain gauge module, built with a precisely calibrated Wheatstone bridge and dual LM324DR amplifiers, successfully converted mechanical strain into measurable electrical signals. This module reliably detected both small and significant changes in strain, thanks to its enhanced signal clarity and sensitivity. The module was tested under various stress conditions, confirming its capability to consistently output proportional changes in voltage with increasing stress, verifying the fundamental relationship.

Our custom-designed voltage regulator module, centered around the AMS1117-5.0 LDO regulator, successfully provided a stable 5V output across a range of input voltages (6.5-12V). This stability was crucial for powering the microcontroller and other sensitive components without interruption.

The STM32F103C8T6 microcontroller acted as the brain of our operation, effectively managing data from the strain gauge and executing control commands for real-time traffic light signaling based on structural data.

The Bluetooth module (RC6621A) enabled seamless wireless communication, allowing for the transmission of structural data and system alerts to mobile devices, facilitating remote monitoring and data analysis.

Traffic Light System successfully responded to varying loads on the model bridge. A remote-controlled car simulated different weights, triggering appropriate traffic light responses (green to red) as structural limits were approached and surpassed.

Using a support vector machine in Python, we developed a predictive model that learned from the strain gauge data to make decisions. This model was simplified and integrated into the microcontroller via C programming, demonstrating the successful blending of data analytics with embedded system responses.

The embedded system's capability to process and react to data in real-time was showcased through continuous monitoring and adaptive signal control based on the analyzed inputs from the structural sensors.

5.2 Uncertainties

Over the course of our project, while most of the results met expectations, there were also areas of uncertainty and where the results did not exactly match our original projections. Here, we provide a quantitative discussion of these unsatisfactory results, clarify the problems encountered, and support our interpretation with evidence from our testing and analysis.

Low voltage regulator performance: When the input voltage drops to the lower end of 6V, the AMS1117-5.0 regulator does not achieve optimal performance. This is evidenced by the deviation of the output voltage, which is lower than the expected 5V threshold and can be as low as 4.7V.

After analysis, it was observed during the test that when the input voltage was set to 5V, the stability of the output voltage decreased, and the standard deviation of the output voltage was ±0.3V, while the standard deviation of the output voltage under normal working conditions (6.5-12V) was ±0.05V. This also means that the low difference (LDO) characteristic of the regulator assumes very little difference between the input and output voltages; However, due to inherent inefficiency and a drop voltage characteristic of about 1.2V at full load, it is difficult for the regulator to maintain a stable output when the input is close to the output voltage.

Sensitivity of strain gauges to environmental changes: Strain gauges exhibit sensitivity to environmental changes, particularly temperature fluctuations, which were not fully anticipated at the design stage. This results in slightly inaccurate strain measurements, with the resistance of the strain gauge changing by about 0.2% for every 1°C change in temperature, a sensitivity that can result in a strain reading error of up to 5%, but does not have a large impact on the experiment as a whole.

5.3 Ethical considerations

Collecting real-time data on vehicles crossing the bridge could potentially trigger privacy concerns, particularly if the gathered information includes identifiable details about individuals or vehicles. This might contravene the principle of upholding privacy as delineated in the IEEE Code of Ethics, which underscores the importance of safeguarding individuals' privacy and confidentiality. To address this, it's imperative to anonymize and aggregate the collected data to prevent the identification of individuals or specific vehicles. Additionally, stringent access controls and encryption protocols should be implemented to protect the integrity of the collected data.

The development process must prioritize transparency regarding the system's capabilities and constraints. It is essential to establish clear accountability for any decisions derived from the collected data. This commitment resonates with the IEEE Code of Ethics, underscoring the significance of honesty and integrity in professional endeavors. It is imperative to comprehensively document the development process, encompassing the algorithms employed for data analysis and the criteria guiding alert generation. Stakeholders should receive lucid explanations about the system's functionality and its potential ramifications.

When assessing the force dynamics of the physical model of the bridge, an idealized model is employed, which simplifies by neglecting the impact of the self-weight of the steel plates situated both within and outside the piers serving as supports. This oversight extends to the deflection of the bridge. Additionally, the model fails to consider the width of the piers in relation to the bridge's longitudinal direction, contributing to discrepancies between simulated bridge data and results from actual experiments. To rectify this, experimental statistics are utilized to amend the relationship between the idealized and actual models. This correction is based on the average deviation between the outcomes derived from the ideal model and those observed in reality. Notably, while the maximum transverse shear force of aluminum 7075 significantly exceeds the maximum shear force calculated via the ideal model, this discrepancy in transverse shear force does not adversely impact the experimental findings.

Since the maximum weight limit for triggering the system to display a red light is set with respect to the next bridge, and in reality, routes between bridges typically aren't singular, this could potentially trigger unnecessary alerts.

5.4 Future work

For the future development of this project, we have several ideas for improvement.

One potential improvement is to expand the range of sensor types to include environmental types such as temperature, wind and humidity, which would provide a more comprehensive understanding of the factors that affect the structural health of Bridges.

In addition, integrating advanced machine learning algorithms can improve the accuracy of predicting potential structural failures, so that proactive maintenance actions can be taken based on real-time data insights, and for feasibility first reasons, we only chose SVM for simple machine learning at design time.

Improving the traffic light communication system to make its interface more user-friendly will facilitate better interaction with stakeholders and ensure that important information is more accessible and actionable.

What’s more, ensuring seamless compatibility with other transportation and infrastructure management systems can make digital twins a key component of the broader smart city framework, thereby improving overall transportation efficiency and safety.

Object recognition is also a very feasible development direction. Machine learning models such as yolo algorithm can be used to transmit real-time effects with cameras on Bridges and identify and locate vehicles through object recognition technology, which is also of great help to the analysis of bridge safety.

By implementing these enhancements, projects can achieve higher levels of efficiency, safety, and proactive management, aligned with future smart infrastructure goals.

References

[1] G. He, S. Yu, H. Wang, C. Liang and S. Yang, "Study on Thermal Compensation Treatment Method of Strain Measured by Resistance Strain Gauge Method in Reflow Soldering Process," 2022 23rd International Conference on Electronic Packaging Technology (ICEPT), Dalian, China, 2022, pp. 1-5, doi: 10.1109/ICEPT56209.2022.9873482. keywords: {Temperature measurement;Temperature;Thermal resistance;Current measurement;Strain measurement;Thermal analysis;Electrical resistance measurement;PCB Assembly;Strain Gauge Method;Thermal Compensation;Reflow Soldering}

[2] Y. Liu, F. Xu, Z. Yang, Y. Deng and X. Wang, "Research on the Application of Digital Twin in Bridge Rotation Construction," 2023 6th International Conference on Computer Network, Electronic and Automation (ICCNEA), Xi'an, China, 2023, pp. 419-423, doi: 10.1109/ICCNEA60107.2023.00094. keywords: {Bridges;Wireless communication;Prototypes;Games;Real-time systems;Rail transportation;Digital twins;Digital Twin;OPC UA;Bridge Crane Slewing;Unity3D}

[3] X. Xue, H. Zhu, Y. Chen and Y. Wang, "Research on Construction of Digital Twin System for Bridge Vehicle Load Identification," 2022 8th International Conference on Hydraulic and Civil Engineering: Deep Space Intelligent Development and Utilization Forum (ICHCE), Xi'an, China, 2022, pp. 536-539, doi: 10.1109/ICHCE57331.2022.10042557. keywords: {Bridges;Deep learning;Image recognition;Target tracking;Heuristic algorithms;Cameras;Real-time systems;bridge engineering;vehicle load positioning;digital twinning;virtual data set}

[4] Y. Yu, C.S. Cai, L. Deng

State-of-the-art review on bridge weigh-in-motion technology

Adv. Struct. Eng., 19 (2016), pp. 1514-1530, 10.1177/1369433216655922

[5] Haibing Wu, Xin Wu, Wei He, Fang Wang, Lu Deng,

A computer vision-assisted method for identifying wheel loads of moving vehicles from dynamic bridge responses, Mechanical Systems and Signal Processing, Volume 197,2023,110372,ISSN 0888-3270,

https://doi.org/10.1016/j.ymssp.2023.110372.

[6] X. Lai, Mustafa Furkan, I. Bartoli, A. Emin Aktan, and K. Grimmelsman, “Bridge weigh-in-motion using augmented Kalman filter and model updating,” Journal of civil structural health monitoring, vol. 12, no. 3, pp. 593–610, Mar. 2022, doi: https://doi.org/10.1007/s13349-022-00559-3.

Appendix A Requirement and Verification Table

Table 2 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Voltage regulator requirement a. When the input voltage is 6.5-12V, the output voltage is 5V±0.05V.	1. Verification a. Measurements are made in the laboratory using generators and oscilloscopes.	
2. Strain gauge requirement a. When the strain gauge is bent in two directions, the resistance increases and decreases respectively. b. When the resistance value is small, the output voltage becomes smaller. c. The output voltage range is 0-5V.	2. Verification a. Use a resistance meter to measure. b. Measure the output voltage with an oscilloscope. c. Increase the bend in the range of 30 degrees and ensure that the output voltage is always below 5V.	
3. Controllerc requirement a. Must process sensor data in real-time with a latency of no more than 10ms. b. The traffic light system works according to the design logic.	3. Verification a. Verify that the error time is within the allowable range. b. Check traffic light status.	
4. Bluetooth requirement a. Must update the digital model in real-time with a maximum delay of 100ms from data receipt. b. Must simulate bridge behavior with an accuracy of 98% compared to the physical model. c. Must be capable of running predictive algorithms to forecast potential structural issues.	4. Verification a. Verify that the error time is within the allowable range. b. Confirm errors in pressure signal and strain gauge signal. c. Check the running status of the algorithm.	