14th Asia-Pacific-Euro Summer School on Smart Structures Technology, Hong Kong, China



LAB experiment

**Design and Implementation of Wireless IoT Sensors**

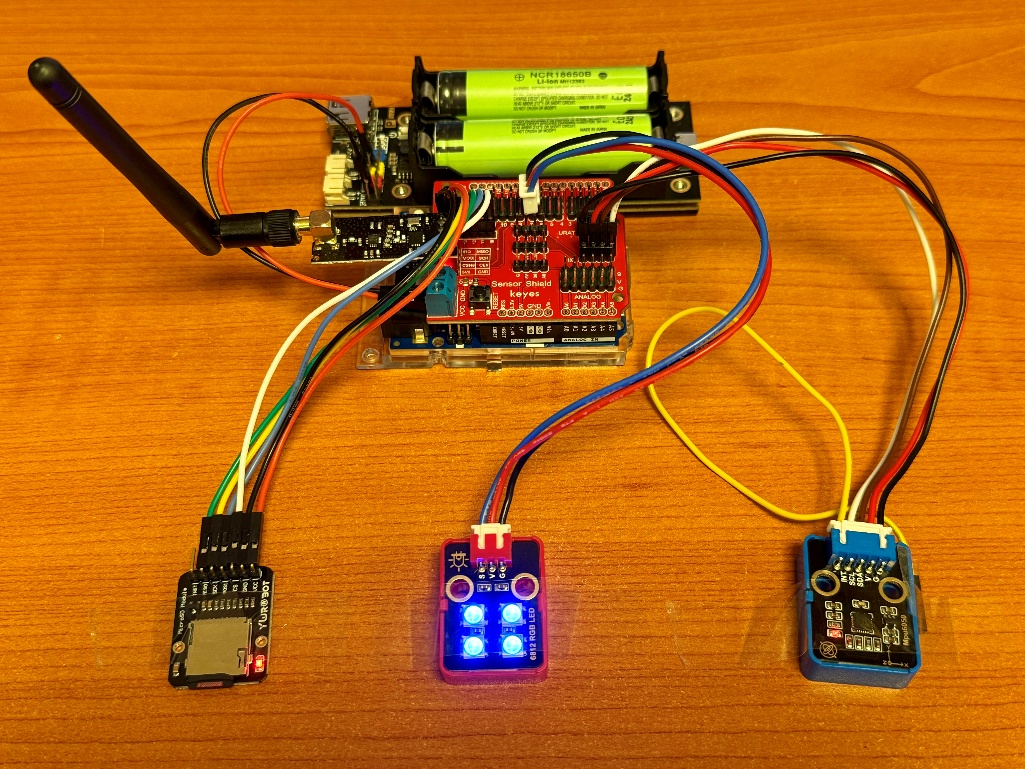
|  |  |
| --- | --- |
| Time: | 29-July, 14:00 - 17:15 |
| Location: | TU103 + ZS1107 |
| Instructor: | Yuguang Fu,  Nanyang Technological University |

# Description

This experiment aims to demonstrate how to build and use a simple wireless IoT sensor/network using Arduino Uno R4 Wifi, including (1) hardware setup, (2) programming, and structural vibration tests.

# Hardware Setup

An IoT wireless sensor should have basic abilities of data collection, storage, communication, and computation. These functions depend on relevant hardware components, which are described in the following.



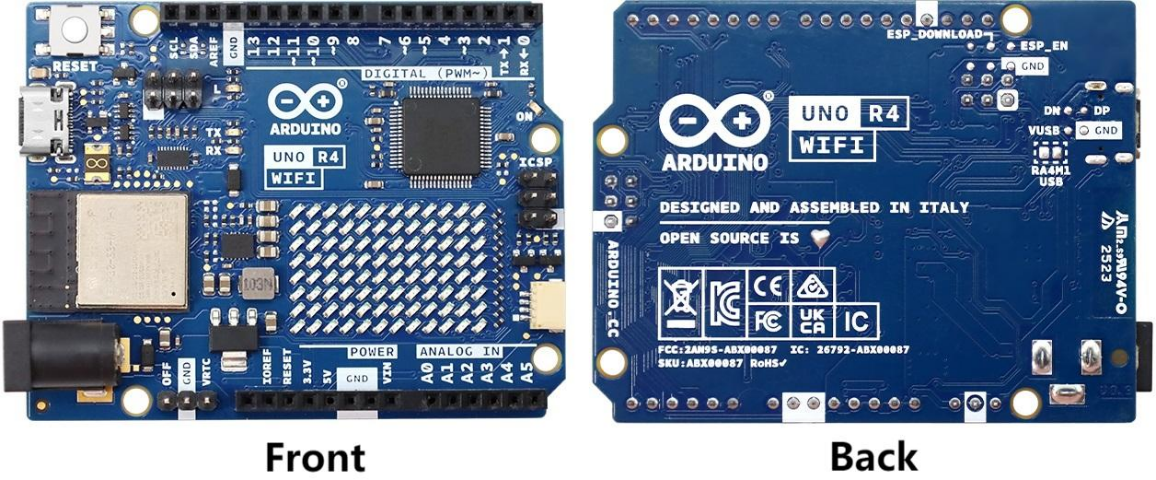
**Figure 1.** Prototype for an IoT wireless sensor

**Table 1.** Item list

|  |  |
| --- | --- |
| **Component** | **Description** |
| Arduino UNO R4 WIFI | Main controller, providing processing power and connectivity. |
| Sensor Shield | A shield that connects to the Arduino for easy sensor integration. |
| MPU6050 | An acceleration sensor that measures motion and orientation. |
| SD Module & Card | Used for data storage, allowing the node to log sensor data. |
| RGB LED | Provides visual feedback, indicating the status of the node. |
| NRF24L01 Module & Antenna | Enables local wireless communication between nodes. |
| BMS & Battery | Provides power to the entire node, making it portable. |
| Enclosure | Protects internal electronic components and provides a pretty appearance. |

## 2.1 Main Controller Board

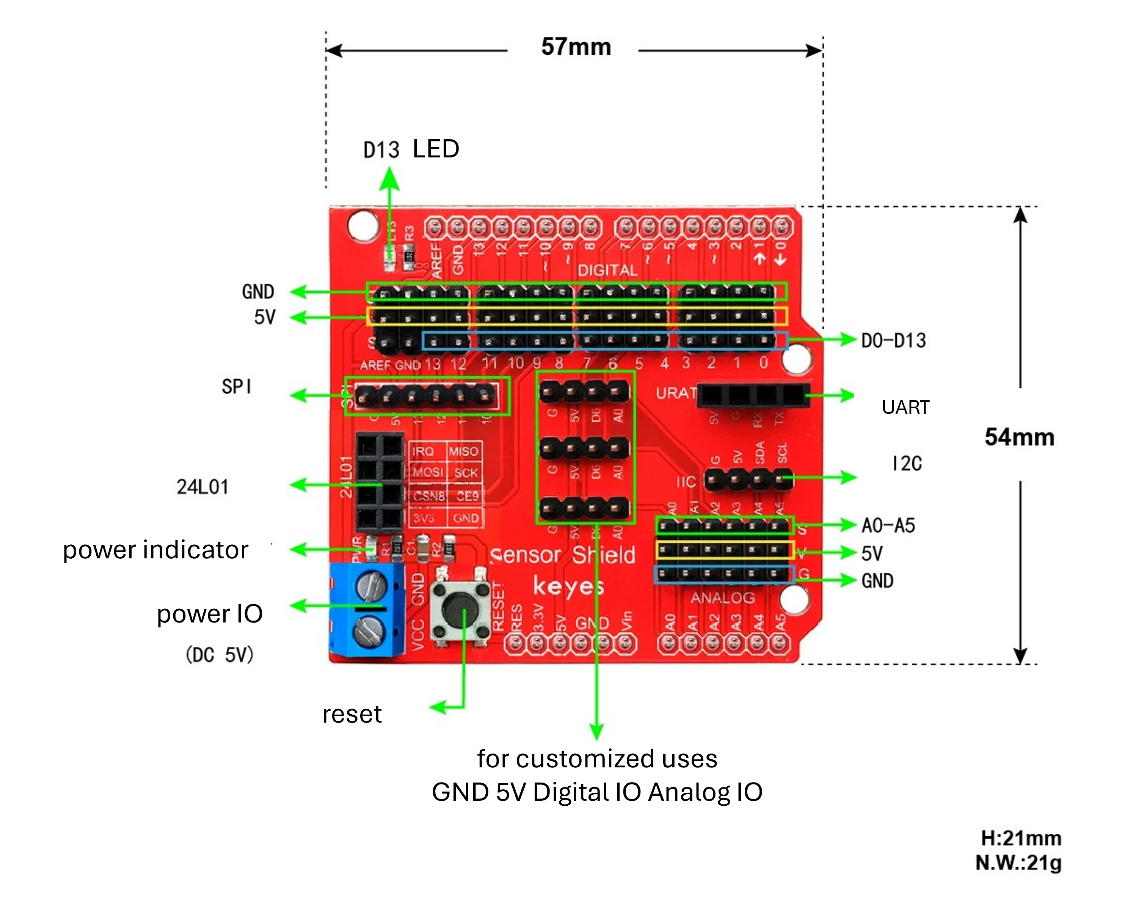
The Arduino Uno R4 WIFI (Figure 2) is designed around the 32-bit microcontroller RA4M1 (Arm® Cortex®-M4) with a 48 MHz clock speed, 32 kB SRAM and 256 kB flash memory. It also features an ESP32-S3 for Wi-Fi®/Bluetooth® connectivity.



**Figure 2.** Arduino Uno R4 WIFI

## 2.2 Sensor Shield

The Sensor Shield (Figure 3) is an expansion board designed for connecting various sensors and other components to an Arduino board, providing a rich set of interfaces to facilitate wiring.



**Figure 3.** Sensor shield

## 2.3 Accelerometer

The MPU6050 (Figure 4) is used for acceleration measurement. In fact, MPU6050is an inertial measurement unit (IMU) that integrates a 3-axis gyroscope and a 3-axis accelerometer. It is widely used in various applications such as robotics, drones, and motion tracking. Here, we only use the acceleration components in three axes (X, Y, Z).



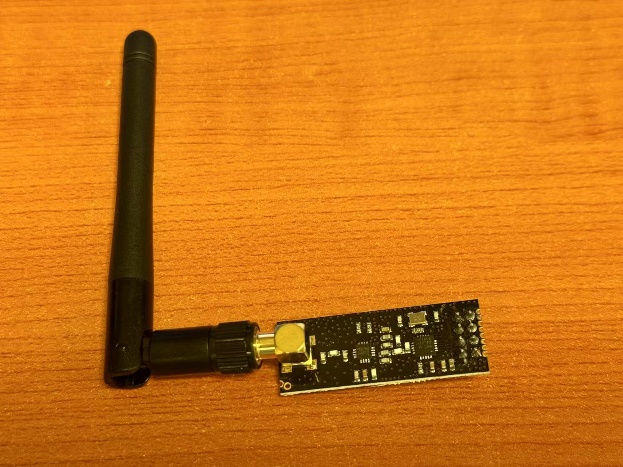
**Figure 4.** MPU6050

**Table 2.** Wiring of MPU6050

|  |  |  |
| --- | --- | --- |
| **Sensor Shield Pin** | **MPU6050 Pin** |  |
| VCC | VCC |
| GND | GND |
| SDA | SDA |
| SCL | SCL |
| **-** | INT (optional) |

## 2.4 Radio Frequency

Radio frequency (RF) module nRF24L01 is a low-power, low-cost 2.4GHz wireless transceiver suitable for short-range wireless communication. It supports multiple data rates and multi-channel operation, has strong anti-interference capabilities, and offers a longer transmission distance. Compared to WIFI, radio frequency features low latency, low power consumption, and is more suitable for applications requiring real-time data transmission.



**Figure 5.** nRF24L01

**Table 3.** Wiring of nRF24L01

|  |  |  |
| --- | --- | --- |
| **Sensor Shield Pin** | **nRF24L01 Pin** |  |
| GND | GND |
| 3V3 | VCC |
| CE9 | CE |
| CSN8 | CSN |
| SCK | SCK |
| MOSI | MOSI |
| MISO | MISO |
| IRQ | IRQ (optional) |

## Tip: just plug the module in will do.

## 2.5 RGB LED

RGB LED (Figure 6) is a type of LED that can display multiple colors. It typically consists of three independent LEDs that emit red, green, and blue light. By adjusting the brightness of these three colors, various colors can be mixed.



**Figure 6.** RGB LED

**Table 4.** Wiring of RGB LED

|  |  |  |
| --- | --- | --- |
| **Sensor Shield Pin (D7)** | **RGB LED Pin** |  |
| V | V |
|  |  |
| G | G |
|  |  |
| S | S |

## Tip: please use the column marked by number 7 for consistency with the code.

## 2.6 SD Card Module

The SD card module is an external storage device used for data storage. It communicates with Arduino via the SPI interface. The SD card module is typically used to store log data, configuration files, or other data that needs to be persistent.



**Figure 7.** SD card module

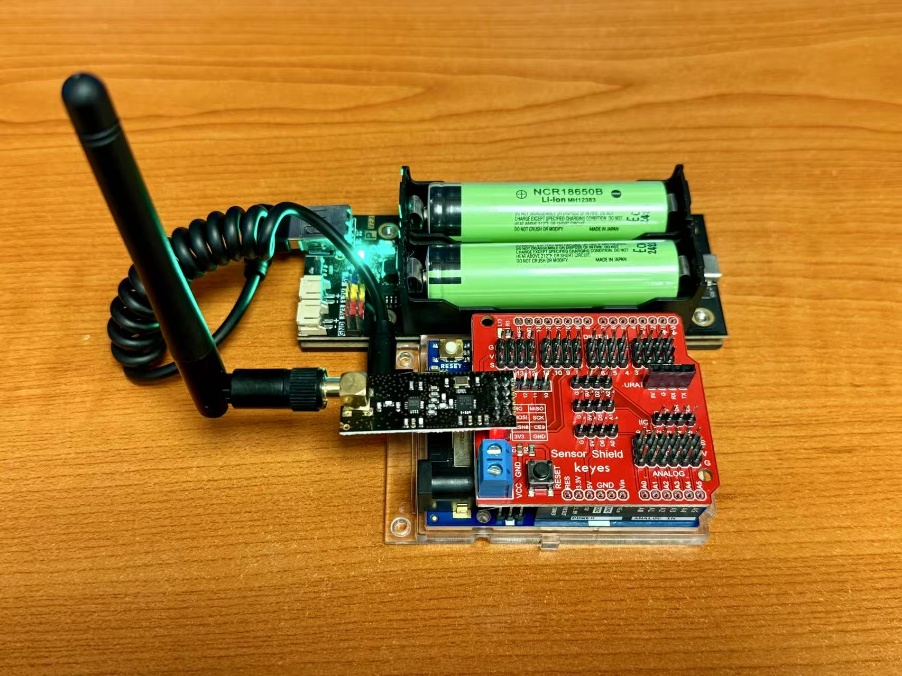
**Table 5.** Wiring of SD card module

|  |  |  |
| --- | --- | --- |
| **Sensor Shield Pin** | **SD card module Pin** |  |
| 5V | VCC |
| GND | GND |
| D10 | CS |
| D11 | MOSI |
| D12 | MISO |
| D13 | SCK |

## Tip: please ensure that the Dupont wires are connected tightly.

## 2.7 Power Supply

Here uses two 18650 lithium batteries (Figure 8) for the power supply. The 18650 lithium battery is a commonly used rechargeable battery with high energy density and long service life. They are typically used in portable electronic devices and power tools.



**Figure 8.** Batterypower

To facilitate the installation and replacement of batteries, this project uses a battery holder. The battery holder can accommodate two 18650 lithium batteries and provides the necessary connection interface. It is usually equipped with an indicator light, allowing users to check the battery status easily. Please note that the batteries may not fit perfectly into the holder—this is not an issue. Do not apply excessive force. As long as the power indicator lights up, the connection is sufficient.

## 2.8 Enclosure

An enclosure is designed to house various components. The enclosure is constructed from durable materials to ensure longevity and protection for the internal components.

|  |  |
| --- | --- |
|  |  |
|  |

**Figure 9.** Different views of enclosure

## 2.9 Sensor Assembly

Assemble an IoT wireless sensor node according to the following steps:

|  |
| --- |
| 1. Connect the Arduino Uno R4 WIFI with the Sensor Shield (main board). 2. Put batteries in the holder. 3. Put the main board and the battery holder into the enclosure (see Figure 9). 4. Connect the MPU6050 to the Sensor Shield (see Table 2), with wires passing through the hole. 5. Connect the radio frequency nRF24L01 to the Sensor Shield (see Table 3), with the antenna passing through the hole. 6. Connect the RGB LED to the Sensor Shield (see Table 4), with wires passing through the hole. 7. Connect the SD card module to the Sensor Shield (see Table 5) and put it inside the enclosure. 8. Cover the enclosure. 9. When in use, connect the battery to the Arduino Uno R4 WIFI via USB. |

# Programming

## Install Development Environment

For programming, the first thing is to build the development environment. Here, we chose Visual Studio Code (VSCode) as the code editor. To facilitate the development, a VSCode plugin, PlatformIO, is required to be installed. It supports embedded project management, debugging, code build, etc.

Install the development environment according to the following instructions.

|  |
| --- |
| 1. Install VS Code  * Download and install VSCode from <https://code.visualstudio.com/download>  1. Install PlatformIO  * Open VSCode Extension Manager, or press Ctrl + Shift + X * Search for official PlatformIO IDE extension * Install PlatformIO IDE   VSCode Extensions Manager and PlatformIO IDE auto-installer |

## Software Architecture

Figure 10 shows the dependency graph of the software architecture. To be friendly to reuse and maintenance, the code is modular and well-separated according to different functions. The ‘main.cpp’ is the entrance program, which incorporates different components together.

**Process of Code Build:**

* Preprocessing (.hpp/.cpp files)

The preprocessor expands macros and includes, preparing the code for compilation.

* Compilation (compile each .cpp file separately)

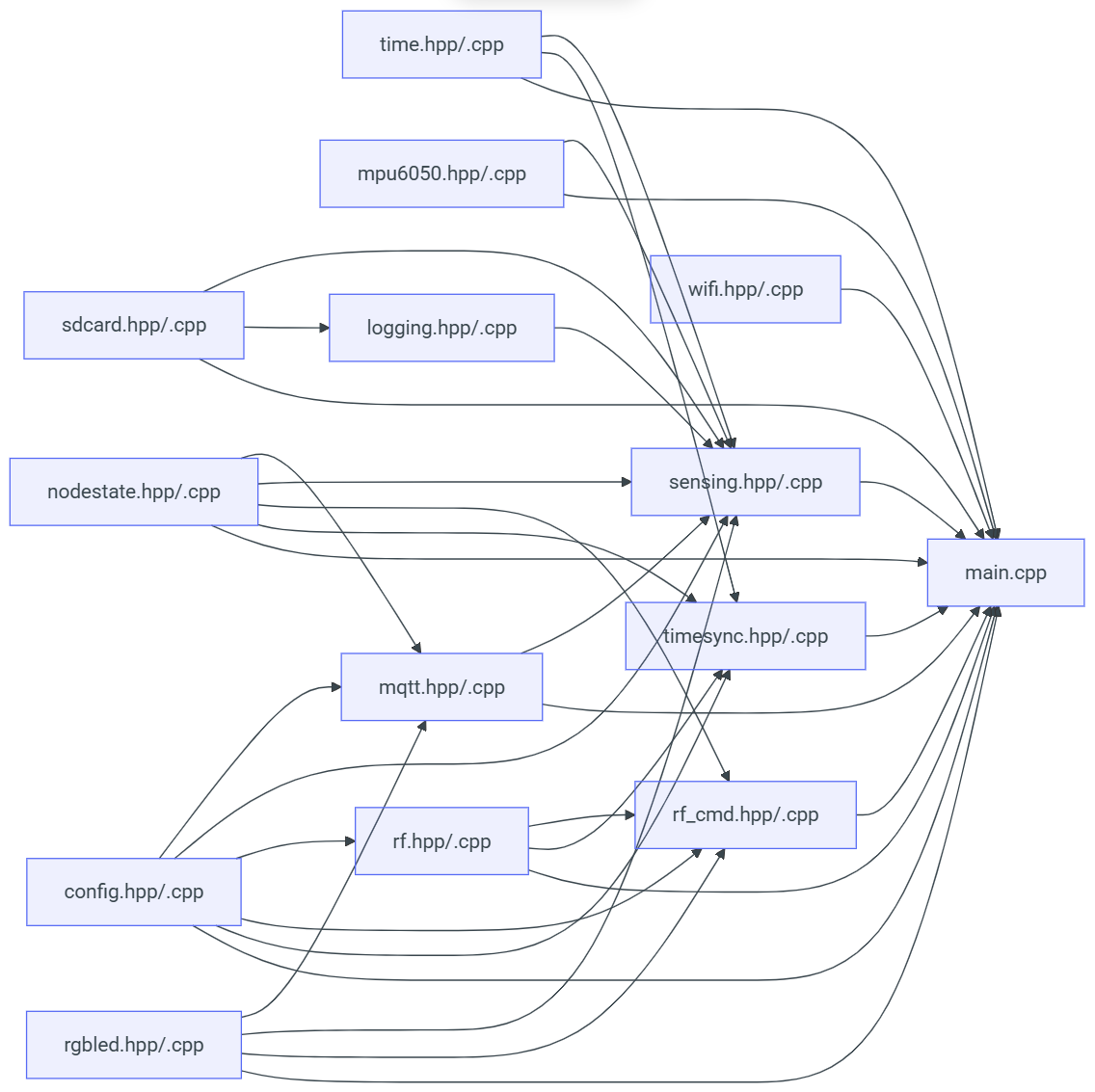
Each .cpp file is compiled into an object file (.o or .obj) independently.

* Linking (combine .o files into an executable)

Once all .cpp files are compiled to object files, the linker combines them into a single executable.

* Uploading (Flashing)

The compiled and linked code is transferred to the embedded system's memory. Once power is on/reset, the code is then executed on the embedded system.



**Figure 10.** Software architecture

## Source Code Modification

The source code (the files shown in Figure 10) is provided. Please download the source code to your local PC and modify the configuration file ‘config.hpp’ accordingly.

Create a new project according to the following steps:

|  |
| --- |
| 1. Download the source code from GitHub  * Open the following URL with a browser, click Download ZIP, and unzip to your\_path * Or if you prefer git, download the code using the command   git clone https://github.com/Shuaiwen-Cui/ArduinoNode.git   1. Create a new project with PlatformIO  * Open VSCode * Click the PlatformIO icon -> Create New Project -> Import Arduino Project      * In the opened dialog, select the board as Arduino Uno R4 WiFi (you can type for filtering) -> navigate to your\_path that contains the source code -> click Import |

Modify the ‘config.hpp’ file for configuration (there will be a main node and 4 leaf nodes, each requiring a specific configuration):

Turn on the hotspot on your mobile phone to provide a network. Set and remember the network name and password.

|  |
| --- |
| 1. Get a copy of the file config.hpp, name it as “config.gateway.hpp” 2. Open the file “config.gateway.hpp”, modify  * #define GATEWAY * #define NODE\_ID 100 * #define NUM\_NODES <actual number of nodes> * #define WIFI\_SSID <wifi name> * #define WIFI\_PASSWORD <wifi password> * #define MQTT\_CLIENT\_ID "GATEWAY"  1. Get 4 copies of “config.gateway.hpp”, name them as “config.leafnode1.hpp”, “config.leafnode2.hpp”, “config.leafnode3.hpp”, “config.leafnode4.hpp” 2. Open the file “config.leafnode<id>.hpp” (<id>=1,2,3,4), modify  * #define LEAFNODE * #define NODE\_ID <id> * #define MQTT\_CLIENT\_ID "LEAFNODE<id>" |

## Code Compile and Upload

The next step is to build and upload the code to the Arduino Uno R4 board for execution.

|  |
| --- |
| 1. Build and upload code for the main node  * Copy the contents in “config.gateway.hpp” to cover all in “config.hpp” and save (Ctrl + S) * Click the build button () on the status bar at the bottom of VSCode, wait for the build to succeed * Connect one sensor node to your computer via USB, and click the upload button () on the right of the build button, wait for the upload to finish.  1. Build and upload code for the other 4 leaf nodes, leafnode<id> (<id>=1,2,3,4)  * Copy the contents in “config.leafnode<id>.hpp” to cover all in “config.hpp” and save (Ctrl + S) * Click the build button () on the status bar at the bottom of VSCode, wait for the build to succeed * Connect another sensor node to your computer via USB, and click the upload button () on the right of the build button, wait for the upload to finish |

# Experiments

The experiment aims to employ the assembled wireless sensors to collect structural vibration signals. The collected signal can be used for structural condition assessment, modal parameter identification, and so on.

## Lab test

## Footbridge test

* Place and fix the 5 sensor nodes at 1/6L, 2/6L, 3/6L, 4/6L, and 5/6L of the footbridge.
* Power all sensors, and use the miniprogram to send a command to the main node to start data sampling. The format of the command is “xxxxxxxxxxxx”.
* Once the LED turns yellow, excite the bridge by jumping and wait for the data collection to complete.
* The collected data will be stored on the SD card, which can be read into your computer for viewing and processing.



**Figure x**. Footbridge

# Conclusion

This document offers step-by-step instructions for assembling an IoT wireless sensor using various components, modifying and uploading code to the Arduino board, and applying the sensor prototype for vibration data collection.