Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

MASTER FINAL THESIS

Using Low-cost IoT-based Inclinometers for Damage Detection of a Bridge Model

Final Thesis developed by:
AHMAD ALAHMAD

Directed by:
José Turmo Coderque
Seyedmilad Komarizadehasl

Master in:
Structural and Construction Engineering

Barcelona, 27/09/2022

Department of Civil and Environmental Engineering

Acknowledgment

Without the help of many, this work would not have been accomplished. Words cannot express my gratitude to Prof. José Turmo Coderque and Prof. Seyedmilad Komarizadehasl, my advisors, who generously provided knowledge, expertise and support.

I would like to thank all the Secretaria d'Igualtat, Migracions i Ciutadania members for their assistance and support throughout my study. Thanks must also go to the department professors, research assistants, and study participants from the university, who impacted and inspired me.

Lastly, I am so grateful to my parents for their nonstop love, support, and encouragement. Their faith in me has held my spirits and motivation strong throughout this journey.

Abstract

Nowadays, researchers are paying close attention to using inclinometers for Structural Health Monitoring (SHM) applications. Moreover, the applications based on using inclinometers can detect the magnitude and location of bridge pathologies. However, as these applications are based on expensive commercial inclinometers, their use is typically exclusive to the SHM of structures with a high monitoring budget. There is a gap in the literature with the development and validation of low-cost accurate angular-meters for decreasing the monitoring cost of inclinometer-based damage detection applications.

This work aims to develop low-cost IoT-based inclinometers for detecting damage in bridge structures. The Low-cost Adaptable Reliable Angle-meter (LARA) is a novel inclinometer that accurately measures an induced inclination by combining the measurements of five gyroscopes and five accelerometers. The accuracy, resolution, Allan variance, and standard deviation of LARA are examined through laboratory experiments and are compared with those obtained by numerical slope calculations and a commercial inclinometer (HI-INC).

For further experimental validation, a robotic vehicle model is designed and developed to simulate a moving load over a bridge model. The vehicle model integrates IoT technology and can be utilized in different damage detection experiments.

The outcomes of a load test experiment using a simple beam model demonstrate the high accuracy (0.003 degrees) of LARA measurements. LARA may be used for structural damage identification and location in bridges utilizing inclinometers because of its low cost and high accuracy.

Index

Table Index

Figure Index

1. Introduction

In this study, the development of the Low-cost Adaptable Reliable Angle-meter (LARA) is presented as an accurate and cost-efficient inclinometer in the field of structural health monitoring. Since most of the existing commercial inclinometers are expensive and have different limitations, the development of the proposed inclinometer was carried out as an alternative to overcome the mentioned shortcomings.

For further validation of the proposed inclinometer, an IoT-based robotic vehicle model is introduced to conduct different experiments on a beam model. The vehicle model improves the condition of the experiments and provides a more realistic simulation of the traffic loading.

The aim of the experiment is to detect damage in the bridge model using direct rotation measurements. For this purpose, LARA is utilized to obtain the rotation response of the bridge model due to the moving vehicle model. The vehicle model integrates (IoT) technology and can be controlled using an online server and standalone application written in Python. The mechanical components of the model were designed using Fusion360 and printed using 3D-printing technology.

This study establishes the foundation of a reliable and cost-efficient damage detection system that integrates IoT technology into all its elements. Besides, the study attempts to introduce compatible robotic system which can be utilized to perform different tasks in the SHM process.

1.1. Objectives

The main objective of this study is to develop an economical and reliable inclinometer that can be used in different damage detection applications. The specific objectives behind this work are:

- Design the LARA inclinometer hardware and software.
- Establish data acquisition system.
- Validate the output of the inclinometer.
- Develop the robotic vehicle and the beam model to conduct damage detection experiments using LARA.

1.2. Contents of the work

This work consists of five chapters. First, the introduction is illustrated, and then a comprehensive literature review is presented in the state of art chapter. The third chapter demonstrates the development of LARA, followed by the development of the robotic vehicle model in the fourth chapter. The fifth chapter is dedicated to laboratory experiments verifying the accuracy and resolution of the LARA. Finally, in the last section, the main conclusions of the work are drawn.

1.3. Methodology

In order to achieve the objectives of this work, the following approaches were carried out:

1) Literature review of SHM.
2) Comprehensive literature review of low-cost sensors and Arduino-based systems.
3) Development of the inclinometer sensor.
4) Development of the robotic vehicle model.
5) Development of the simply supported beam model to validate the inclinometer.
6) Laboratory validation of the developed inclinometer.
7) Writing scientific articles and the master's thesis.

2. State of the art:

Engineers have been interested recently in structural health monitoring (SHM) as a control system to monitor the structural response of structural components to mitigate probable familiars in civil infrastructures in the future. The serviceability and safety of the building may gradually deteriorate due to a variety of circumstances and events such as construction defects, fatigue, and environmental conditions [1], [2], [3]. To reduce future repair costs and ensure structural safety and serviceability, it is crucial to monitor and evaluate the health status of structures throughout their lifetimes [4], [5]. Applications for SHM offer essential details regarding the performance, condition, and real structural response of infrastructures. According to numerous academics (such as [6], [7]), The simulated model of the genuine structures (digital-twin), which replicates the functionality of the structures, may be calibrated using SHM [8]. The decision-making options throughout the maintenance phase of the structures under investigation can then be evaluated using digital twins [9], [10].

Using Structural System Identification (SSI) approaches, SHM data may be utilized to identify structural characteristics [11], [12]. Identification of the structural system aims to determine the characteristics of the computer-based model (such as axial or flexural stiffness) to calculate the structural response of the structure [13]. SSI approaches can be categorized as static or dynamic depending on the characteristics of the structural reaction and the external stimulation [12].

Temperature and humidity-related environmental events might lead to structural diseases such cracks opening [14], rotations, settlements, and corrosion [2], [15], [16]. These changes develop very slowly over time. As a result, they are categorized as static. Dynamic structural reactions, on the other hand, are brought on by excitations that alter immediately over time, such as vibrations caused by traffic, ambient activities, and seismic activity waves [17].

Sensors are often utilized in SHM systems to measure static and dynamic reactions [18]. Accelerometers are frequently employed to track the dynamic behavior of structures, whereas strain gauges, clinometers, and thermometers are the most frequently used sensors for static measurements [19], [20].

By detecting changes in the structural response, accelerometers may estimate and identify a structure's dynamic features [21], [2], [22]. Even though accelerometers may detect global structural damage to a structure, they typically fail to determine the location and degree of the damage [23]. As long as a specific reference point exists, displacement sensors such as Laser Displacement Sensors (LDS) can be utilized in load testing to assist identify the damage and its extension [24], [25]. Unfortunately, there are a few restrictions on-site that might make it challenging to define the necessary reference points [26]. Alternative strain-type sensors can be employed to assess the damage's location and amount. In the literature, this kind of sensor has demonstrated very good accuracy and applicability [27], [28], [29]. To completely monitor the structural features of a structure, however, many sensors of this type may be required [30].

Inclinometers can be utilized to get around the shortcomings of the aforementioned sensors. A target object's angular rotation with respect to an artificial horizon can be estimated using angular sensors (inclinometers, tilt sensors) [23]. The majority of inclinometers operate on the concept of monitoring reactions generated by gravitational pendulum behavior [31]. The drift of vertical members and the vertical deflection of the horizontal parts may also be calculated using this slope [32].

In recent decades, inclinometer sensors have found widespread application in a variety of industries. Inclinometers were initially used for geotechnical purposes in the civil engineering sector. As sensor precision has increased over time, it has become possible to apply it to other areas of civil engineering, such as monitoring the structural health of bridges [31]. The structural response of bridges has been extensively studied in the literature using inclinometers. In order to examine a post-tensioned concrete bridge during its construction, post-tensioning, and first-year operation stages, Glii

et al. [33] used long-gauge deformation sensors and inclinometers. The results were utilized to verify the post-tensioning and overall health of the bridge.

Inclinometers play a significant role in the long-term monitoring of long-span (such as cable-stayed and suspension) bridges [34], according to literature reviews [35], [36], [37]. Additionally, a number of researchers (including [38], [39], [40]) noted the conventional use of inclinometers while examining the bridge abutments' response to boundary conditions. These sensors are also frequently used to calculate the deflections of bridge decks [41], [42], [43].

Table 1 illustrates the characteristics of some of the commercially available inclinometers and is sorted by the price of the sensors. This information includes the measurement range, the resolution, the sampling rate and the cost of the introduced inclinometers. It should be noted that prices are based on the recent producer declaration and are VAT excluded.

Table 1 Characteristics of some of the commercially available inclinometers.

MODEL	MEASUREMENT RANGE (DEGREES)	RESOLUTION (DEGREES)	SAMPLING RATE (HZ)	PRICE (€)
ZEROTRONIC	± 0.5°	100×10-5 °	10	3950
JDI 200	± 1.0°	10×10-5 °	125	2250
T935	± 1.0°	6×10-5 °	10	1696
ACA2200	± 0.5°	10×10-5 °	20	710
HI-INC	± 15.0°	100×10-5 °	100	650
ZCT-CX09	± 15.0°	100×10-5 °	8	350
DNS	± 85.0°	300×10-5 °	100	348

Table 1 analysis reveals a broad variety of pricing (ranging from 350€ to 3950 €) and measurement ranges (varying between 0.5 and 85.0 degrees). It is clear that inclinometers with a lower range have a higher resolution and cost. Furthermore, high-resolution inclinometers often have higher prices and lower sampling frequencies. In reality, several of the models discussed above have been applied in bridge structural response applications. Examples of sensor applications in Table 1 include the employment of a Zerotronic inclinometer to validate modal calibration procedures with measured data from Switzerland's Lutrive bridge [44].

Despite the benefits of using inclinometers [23], there are not many cases of this monitoring system in the bridge SHM literature [45], [46]. Huseynov [31] cites the lack of sensor technology, low-frequency sampling, and the expensive cost of current inclinometers as some of the reasons why inclinometers are not employed.

To address the inclinometer's aforementioned issue, low-cost sensors can be used. Micro-Electro-Mechanical Systems (MEMS) accelerometers have genuinely revolutionized the measuring field due to their smaller size and reduced cost. For example, the sine of an accelerometer's tilt angle may be calculated by dividing the recorded acceleration by the earth's gravitational force.

However, the resolution of the titling measurement made using this method is not very good. A rapid movement or vibration to the structure will cause a significant acquired inclination. MEMS sensors are frequently paired with gyroscopes to tackle this problem. MEMS Gyroscopes use Coriolis acceleration to measure angular rate, which allows them to detect rotational speed [47].

The fundamental disadvantage of gyroscopes is bias instability, often known as flicker noise [48]. The measurement of bias instability over time while running at a constant temperature is known as bias instability. This drift is caused by the circuit's inherent noises and the imperfections of the

components. The bias instability problem can be solved by varying how the accelerometer's computed inclination is coupled with the gyroscope. The accelerometer's measurements rectify the gyroscope's continual drifting in this case. Almost all modern MEMS inclinometers employ sensor fusion to overcome the shortcomings of the accelerometer and gyroscope. Furthermore, the gyroscope measurements are used to reduce the detrimental effects of a quick movement of the accelerometer estimations [49].

The noises in time-domain series that drift over time are characterized and analyzed using the Allan variance [50].

These days, linking the estimation of an accelerometer and a gyroscope is being employed in more and more research and techniques in the literature to improve the resolution of tilting measurement utilizing MEMS sensors [32]. The Kalman filter and complementary filter are two of the most used techniques. A complementary filter, on the other hand, averages estimated angles from an accelerometer and gyroscope with various weights.

On the other hand, the Kalman filter has a high computing need, according to several researchers [51], [52]. The sampling frequency of an inclinometer employing the Kalman filter is therefore often smaller than that of an inclinometer using the complementary filter [49], [53].

Additionally, it should be emphasized that MEMS circuits are built on sensitive sensors that measure an environmental change that is time-related [2]. In other words, a MEMS circuit's incorporated gyroscope and accelerometer are dynamic sensors. Dynamic noises are a part of any dynamic sensor.

Microcontrollers are commonly used in modern technology to control low-cost sensors. There are many different types of microcontrollers available, with Arduino being one of the most popular. It is built on open-source hardware and software [54].

Some examples of these works include:

1. Yan et al. [55], introduced a low-cost wireless inclinometer with a sampling frequency of 20 Hz and claimed resolution of 0.0025°, delivering data up to 2000 meters away to acquisition equipment. This technology is designed to track the movement of large-scale constructions.
2. Ruzza et al. [53], provide a low-cost inclinometer based on Arduino technology and MEMS circuits with an RMS inaccuracy of 0.162 to 0.304°.
3. Andò et al. [56], presented a low-cost multi-sensor system for investigating building structural reactions. This system is based on Arduino technology and communicates wirelessly over the XBEE interface.
4. Hoang et al. [57], produced an extremely efficient, reliable orientation system for inclinometers in static and dynamic applications. Both the static and dynamic testing had indicated RMS errors of 0.106 and 0.091 degrees, respectively.
5. Khan et al. [58], demonstrated a low-cost inclinometer with a moveable electrode. Inside a parallel plate capacitor, the moveable electrode functions as a pendulum. This inclinometer's resolution is listed as 0.38 degrees [58].
6. Woon Ha et al. [59], presented a low-cost wireless MEMS inclinometer with a measuring inaccuracy of 0.04 degrees for a 0.44-degree inclination. The purpose of this inclinometer is to calculate ground movement.

However, there are a few downsides to the Arduino technology, including: (1) Cost: Although these microcontrollers have not recently undergone updates or improvements, their price has not significantly lowered, (2) Internet: Additional components are required to connect an Arduino to the Internet or a hotspot; (3) Small memory size and low CPU speed in the basic low-cost versions of

Arduino products. NodeMCU is a novel microcontroller based on the Internet of Things (IoT) that may be connected to Wi-Fi hotspots and programmed using the Arduino platform, in contrast to Arduino systems [60]. Its capabilities are comparable to those of the Arduino Due. Moreover, its price is a tiny fraction of Arduino Due price. NodeMCU costs 3.95 € [61], while the price of an Arduino Due is at least 36.95 € [62].

The literature review reveals no precise and economical inclinometers based on Arduino or NodeMCU technology that might be employed in bridge SHM due to the distinctive characteristics of this method of monitoring [31]. Numerous scholars have stated (for example, [23]) that the required tilt accuracy must be lower than 0.05 degrees. It has been demonstrated that the movement of a loaded vehicle on a bridge causes a mid-span inclination of 0.2 degrees in a 20-m long simply supported bridge [23]. The existing low-cost inclinometers have a few shortcomings in common, including the following: (1) Building instructions: since these inclinometers are closed hardware, the industry cannot use them to create a low-cost inclinometer; (2) Accuracy: most of them can not provide similar accuracy to the ones shown in Table 1; and (3) Resolution: most low-cost inclinometers with high sampling frequencies have an insufficient resolution for bridges SHM applications. The Low-cost Adaptable Reliable Angle-meter (LARA) system for SHM of bridges is presented in this work for the first time in the literature to overcome these gaps. Based on the studies conducted for this research, LARA is a low-cost wireless inclinometer with an accuracy of 0.003 degrees that uses IoT-based microcontroller (NodeMCU) technology.

In this study, the complementary filter for merging the gyroscope and accelerometer readings of a low-cost MEMS circuit is studied. In this approach, the Kalman filter may be calculated without the requirement for a powerful computer system. As a result, compared to when the Kalman filter is applied, the sampling frequency will be greater. The complementary filter is used by LARA, a MEMS-based device, to link the outputs of the estimated angles from its accelerometer and gyroscope.

This paper also develops a custom-designed Printed Circuit Board (PCB) containing five low-cost aligned MEMS MPU9250 chipsets, each of which includes a gyroscope, an accelerometer, and a magnetometer, in order to construct an inclinometer with higher accuracy, better resolution, and lower noise density. One thing to keep in mind is that MEMS accelerometers are frequently affected by their inherently noisy counterparts [63]. Circuit components (such as resistors and semiconductors) are the source of intrinsic noise [64].

The primary under-study captured signal is left unchanged by averaging the outputs of multiple dynamic sensors. However, the total number of integrated sensors will average out the dynamic disturbances (also known as Intrinsic noise) that are inherent to the sensors. Smaller dynamic changes, such as acceleration or angular speed, can be noticed when the magnitude of these dynamic disturbances decreases. The Allan variance of several sensor combinations is compared with the results of a single MPU9250 inclinometer to demonstrate the favorable effect of integrating many gyroscopes and accelerometers. Furthermore, laboratory tests are performed, and the findings are compared to those provided by a commercial inclinometer (HI-INC). The results show that sensor combination may make a low-cost sensor resolution similar to a high-quality commercial solution. Finally, LARA is validated by being assigned to a simply supported beam and its estimates are substantiated by theoretical estimation.

3. The development of the system

3.1. Introduction

Structural damage detection using inclinometers is getting wide attention from researchers. However, the high price of inclinometers limits this system to unique structures with a relatively high Structural Health Monitoring (SHM) budget. This paper presents in the first chapter a novel low-cost inclinometer, Low-cost Adaptable Reliable Angle-meter (LARA), that combines five gyroscopes and five accelerometers to measure the inclination. LARA incorporates an Internet of Things (IoT) based microcontroller technology enabling wireless data streaming and free commercial software for data acquisition. This chapter investigates the accuracy, resolution, Allan variance and standard deviation of LARA produced with a different number of combined circuits, including an accelerometer and a gyroscope. To validate the accuracy and resolution of the developed device, its results are compared with those obtained by numerical slope calculations and a commercial inclinometer (HI-INC) in laboratory conditions.

The results of a load test experiment on a simple beam model show the high accuracy of LARA (0.003 degrees). LARA's affordability and high accuracy make it applicable for structural damage detection and locating in bridges using inclinometers.

3.2. Control system description

In the current study BeanDevice® Wilo HI-INC (Figure 1.a), an Ultra-Low-Power (ULP) biaxial WIFI inclinometer, was used as the high-accuracy controlling system. This device contains a built-in data logger that can store up to 5 million data logs with a maximum wireless range of 200 meters. Regarding angle measurements, it combines a high-performance inclinometer sensor and a 24-bit delta-sigma analog-to-digital converter, making it possible to have a high-level accuracy of ±0.003° for ±15° and a resolution of 0.001°. In addition, the body of the HI-INC inclinometer is composed of a lightweight aluminum casing with waterproof capability [65]. The program used for data acquisition is a commercial solution promoted by the BeanDevice company and costs 350 €. By taking in the price of this inclinometer from Table 1 and the needed commercial software for data acquisition, to an account, the whole solution costs around 1000 €.

The data acquisition program of BeanDevice company acquires and in real-time illustrates the X and Y axis' inclinations (Figure 1.b). Finally, it should be noted that the settings of data acquisition (such as sampling frequency) can be modified from the main menu of the commercial program (Figure 1.c).

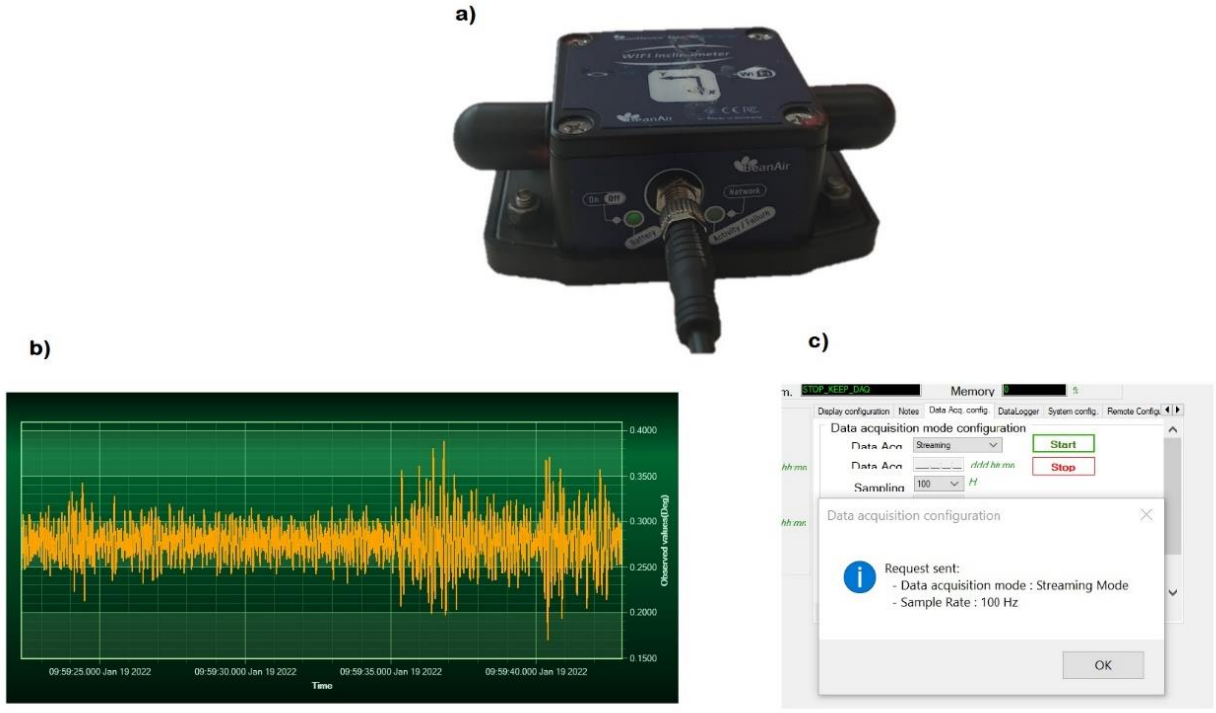


Figure 1 : a) HI-INC biaxial inclinometer, b) inclination streaming over X axis and c) Sampling frequency rate

3.3. Low-cost Adaptable Reliable Angle-meter (LARA) system

In this section, the hardware architecture of the proposed inclinometer is presented. Then, the software part of this system is explained and shown.

3.3.1. Hardware Architecture of LARA

This paper proposes multiple combinations of gyroscopes and accelerometers for producing a more accurate inclinometer. To this end, five chipsets of MPU9250 are engineered together on a single PCB and synchronized using a multiplexor (TCA9548A). To avoid the problems of manual fabrication (such as nonalignment of the circuits, time-consuming process of aligning, soldering and sensor quality control and size), the PCB of LARA was designed and produced to satisfy the delicacy of current project measurements. In addition, the required components of LARA are soldered to the PCB using machine assembly. Figures 2.a and 2.b show the produced sensor and its blueprint. It should be noted that LARA can be assembled by hand using available commercial MPU9250 circuits and a TCA9548A multiplexor. Figure 2.c shows the Fritzing [66] sketch of the system. The cost of a LARA made by connecting five MPU9250 and TCA9548A and a bulk company-produced PCB with assembled components is around 37 and 51 €, respectively.

As shown in Figures 2.a and b, LARA has four output ports. These wires should be connected to a microcontroller to power up the sensors, acquire the sampled data, and convert the gyroscope and the accelerometer to tilt and pitch inclination. The used microcontroller of this paper is NodeMCU and shown in Figure 2.d. This low-cost open-source Internet of Things (IoT) platform runs on the ESP8266 chipset. ESP8266 is a low-cost Wi-Fi microchip with the Internet protocol suite (also known as TCP/IP) capability [67].

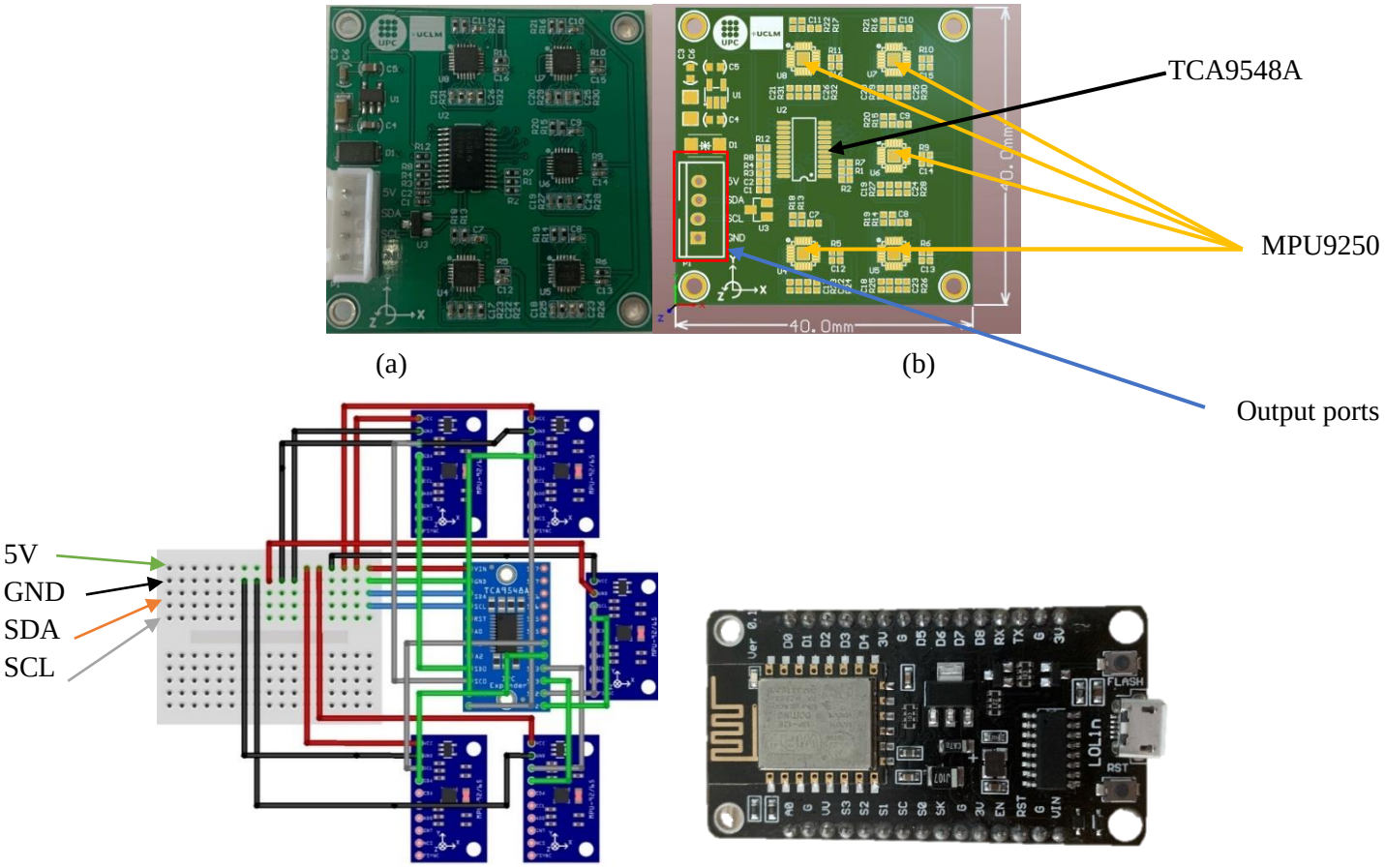


Figure 2 Illustration of LARA: (a) The produced product, (b) The blueprint of the designed PCB, (c) The Fritzing sketch of the system, and (d) NODE MCU microcontroller.

3.3.2. Software Architecture of LARA

In this section, the used software for this project is presented in the following:

Arduino platform: NodeMCU is first programmed using the Arduino platform. This program first estimates the angle in real-time from each of the individual MPU9250 chipsets. Then, the formulas for calculating the rotation using a triaxial accelerometer for X and Y axes are presented in Eq.1 and Eq.2, respectively.

$$angleaccX = \tan^{-1}\left(\frac{accY}{\sqrt{accZ^2+accX^2}}\right)\times\left(\frac{360}{2\pi}\right) \quad (1)$$

$$angleaccY = \tan^{-1}\left(\frac{accX}{\sqrt{accZ^2+accY^2}}\right)\times\left(\frac{360}{2\pi}\right) \quad (2)$$

In Eq.1and Eq.2, where, angleaccX and angleaccY are the calculated angles from the acquired data of a MPU9250 accelerometer around the X-axis and Y-axis, respectively. The accX, accY and accZ represent the obtained acceleration data of X, Y and Z axes. Then, using a complementary filter, the calculated angle from the accelerometers and the acquired data of the gyroscopes are combined. Eq.3 and Eq.4 present the used complementary equation for the fusion of the gyroscope and the accelerometer results for measuring the rotation around X and Y axes, respectively.

$$angleX = \left(0.96\times(angleX_{\,0} + gyroX\times time)\right) + 0.04\times angleaccX \quad (3)$$

$$angleY = \left(0.96\times(angleY_{\,0} + gyroY\times time)\right) + 0.04\times angleaccY \quad (4)$$

In Eq.3 and Eq.4, where, $angleX$ and $angleY$ are the final calculated rotations around X and Y-axes, respectively. The $angleX_0$ and $angleY_0$ are the estimated angle of the system from the previous measurement. In the initiation of the data acquisition, it should be noted that this value equals zero. After that, it represents the rotation progress. The $GyroX$ and $GyroY$ represent the measured angular speed of the gyroscope for X and Y axes, respectively. The $time$ presents the interval time between two measurements. Further analysis of these equations shows that the angle calculated from the accelerometer is multiplied by a smaller coefficient than that of the gyroscope [68]. This low coefficient factor of $angleacc$ is for mitigating the impact of environmental vibrations (also known as cross-talk of vibration) and can vary between 0.02 and 0.05 [69].

These equations are repeated for every MPU9250 chipsets of LARA. Then, the inclination values of the five chipsets are averaged separately for X and Y axes. It is to be known that this code makes the implemented accelerometers and gyroscopes of LARA to sample data and estimate the angles in a synchronized way. Finally, using the already introduced Service Set Identifier (SSID) and the router's password in the Arduino code, the averaged results of X and Y axes are transmitted to a made-up server client by the built-in ESP8266 chipset. LARA prints a server address and a port number at this stage on the serial port of the Arduino. This information should be noted and LARA can be detached from the programming computer. After this, the sensor can be disconnected from the PC and plugged into any available USB power break.

Virtual serial port: After connecting LARA to a USB power source, the data sampling function initiates automatically. This chipset's TCP/IP capability helps this sensor provide its outputs on a local server. A computer connected to the same SSID as LARA can stream the sampled data by introducing the noted server address and port number of LARA. In order to acquire the sampled data and have a real-time graphical representation of the LARA inclination, a virtual serial port application is used [HW [70]]. This free software needs the server address and the port number of LARA and creates a virtual serial port communication connection between LARA and a windows-based computer. By selecting the provided virtual port of the HW software on the Arduino platform, LARA's sampled data can be streamed or graphed just when the sensor is connected to the computer. A computer can indeed be connected physically to several sensors, but with HW virtual serial port, up to 99 devices can be wirelessly attached to a single computer.

Data acquisition: Unlike the Arduino platform, free commercial software like Serial Plot [71] can represent the sampled data in real-time in a graphical interface and save the data with the date and timestamp of data acquisition. The presented flowchart in Figure 3 shows the steps of real-time inclination acquisition using LARA.

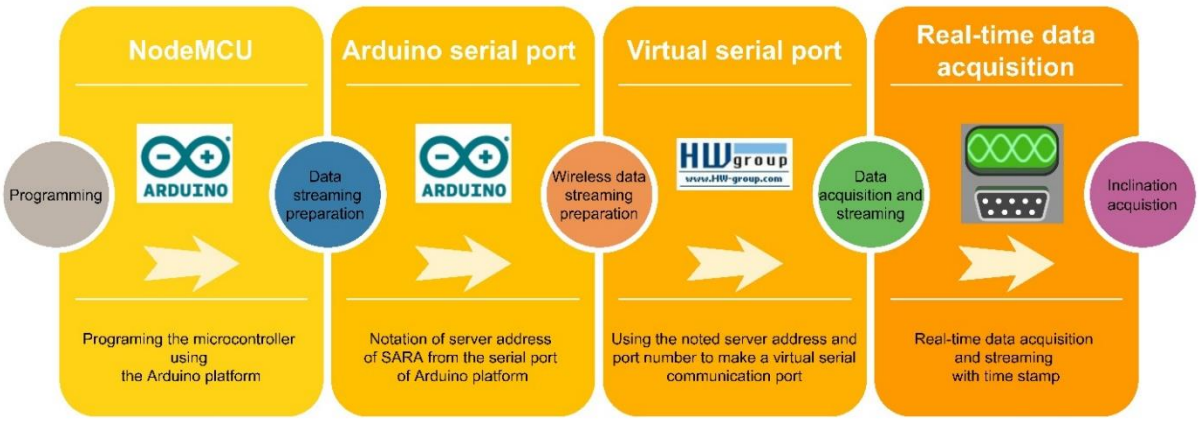


Figure 3 The required steps of real-time wireless inclination acquisition using LARA inclinometer.

3.4. Statistical representation of combining dynamic-sensor theory

This section first studies the effect of sensor combination on the noise density, standard deviation and resolution of angle measurements. Then, the Allan variance and its importance in evaluating the noise density of inclinometers in the literature are explained. Finally, the Allan variances of several combined sensors are presented.

3.4.1. Noise reduction of Inclinometers

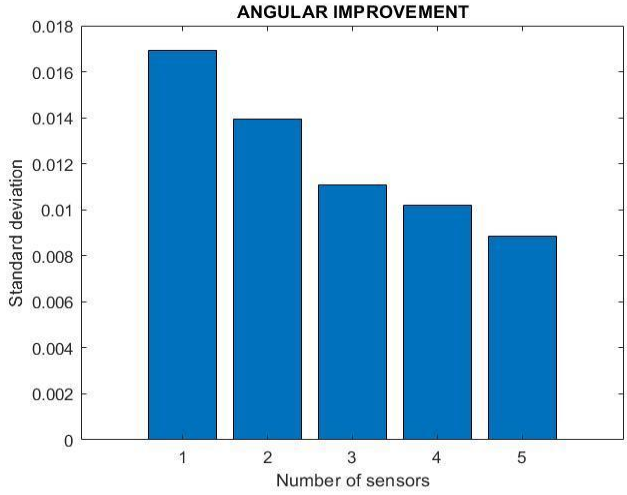
This section explains an experiment that leads to combining up to five similar circuits (MPU9250) for reducing the overall dynamic (harmonic) noises. During this experiment, the inclinometers were placed in a quite environment, far away from crowds and with reduced induced ambient vibrations. The aim of this experiments is to measure and evaluate the pure noise ratio of different combined inclinometers.

It was noticed that the average value of outputs of several aligned synchronized in-clinometers has lower noise density than the those of a single one. The standard deviation of up to five combined inclinometers is presented in Figure 4.a.

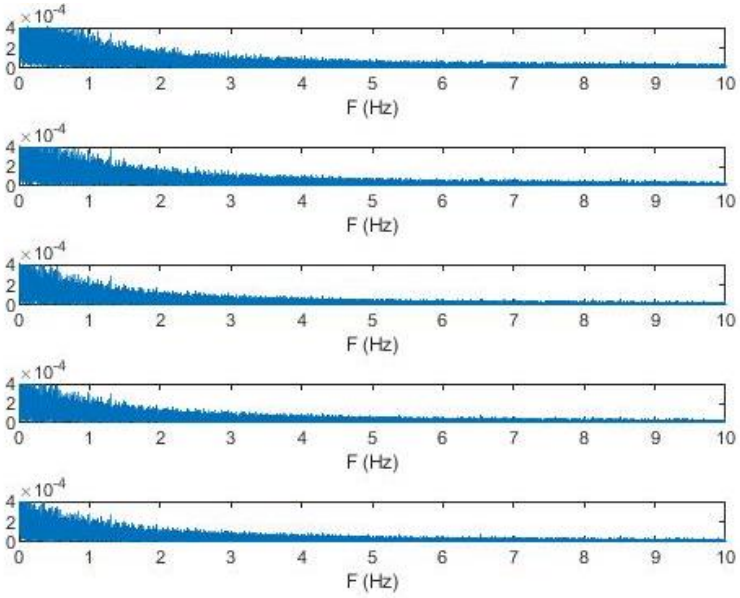
The analysis of Figure 4a shows that the higher the number of sensors considered the lower the noise density of their averaged measurements that the more combined inclinometers have a lower noise density. The reason behind the beneficial behavior of combined inclinometers is within the inherent dynamic noises of the produced accelerometers and gyroscopes chipsets. Figure 4.b shows the frequency domain illustration of the performed experiment. Data transformation from the time domain to frequency domain is done using Fast Fourier transformation (FFT). The analysis of figure 4.b shows that the magnitude of the dynamic noises of the averaged values of a set of sensors made from combined inclinometers is lower than that of a single one. It can be seen that on 1 Hz the measured noises for a single inclinometer and five combined inclinometers are 3.9×10-4 and 2.6×10-4 degrees, respectively.

These results led to investigating the beneficial impact of dynamic sensor combinations. Analyzing the individual outputs, the five used MPU9250 sensors showed that every single sensor has unique dynamic noises.

Furthermore, a single output that includes the averaged inherent noises of all individual inclinometers plus the understudy signal (the rested situation or sets of dynamic movements) is obtained by averaging the outputs of several inclinometers. Since the understudy signals are not dependent on the characteristics of the inclinometers, they have not affected throw-out the FFT process. The FFT highlights the most repeated signals (the understudy ones) and undervalues those that are repeated less, such as the inherent individual noises of the sensors. By improving the noise density, the inclinations that in the first place were smaller than the noise density of the sensor can now be detected due to the improved noise level.



(a)

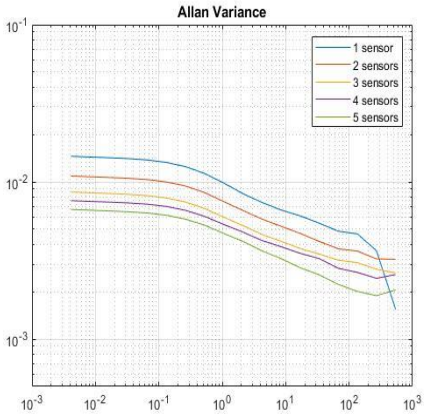


(b)

Figure 4 Representation of the noise ratio of a single and up to five combined inclinometers using: (a) standard deviation, and (b) noise density in frequency-domain.

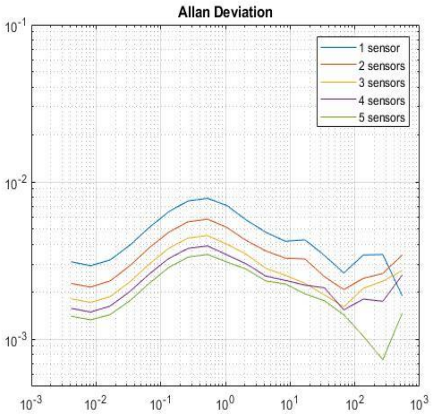
3.4.2. Study of Allan variance

Allan variance is typically used to characterize and analyze those noises that drift throughout time in time-domain series [50]. In fact, Allan variance quantifies the measurement variance of a sensor across different timescales. On the contrary to frequency-domain noise evaluation methods such as spectral Noise Density (ND) [72], Allan variance is a time-domain evaluating tool of different noise sources (such as Quantization, angle random-walk, bias instability, rate random-walk, and rate ramp) [73]. Allan variance shows the progress of a noisy sensor signal over time which can be very useful to identify the progressive random walk of a gyroscope instead of ND that quantifies the noise density of an accelerometer [50]. Allan deviation is more commonly used as the square root of Allan variance [74]. The available acquired inclination acquisition data for measuring the standard deviation of the previous subsection was used for the Allan variance and deviation calculations. Figure 5.a and 5.b shows the log-log plot of Allan variance and Allan deviation of a single and up to five synchronized inclinometers, respectively.



CORRELATION TIME, Log τ (s)

(a)



CORRELATION TIME, Log τ (s)

(b)

Figure 5 Quantifying the noise progress of various inclinometer combinations in time-domain using: (a) Allan variance, and (b) Allan deviation.

Analysis of Figure 5.a shows that the higher number of combined inclinometers, the lower progressive the noise is. For example, the first calculated value of Allan variance (Figure 5.a) of a single inclinometer and five combined ones are 0.0145 and 0.0067, respectively. The beneficial effect of additional synchronized sensors can also be seen in the Allan variation presented in Figure 5.b. It is indicated in the literature [75] that various noise types (such as White noise, Flicker noise and Random) can be detected from the log Allan deviation plot. Detection of different noise types from a gyroscope output is presented in [76].

This section showed that sensor combination decreases noise magnitude in both time-domain and frequency-domain.

4. Development of IoT based robotic vehicle model

The integration of IoT technology has modernized the manufacturing industry through using embedded systems with microcontroller and sensors. These embedded systems can reduce the physical interaction to control devices, optimize time losses and improve efficiency.

The main purpose of developing the proposed vehicle model was to introduce more realistic simulation of a moving load in the validation experiment. However, the model has the core of any IoT based device with the potential for further improvement to perform different type of tasks. The vehicle model help with the automation of the experiments in laboratory, it can communicate with other devices included in the experiments and increase the accuracy of the experiment.

In this section, the components of the vehicle model are presented. Then, the software to configure the model is explained and shown.

4.1. The mechanical component

The mechanical parts were designed using Fusion360 software, the vehicle model consists of two axles 10cm spaced. The distance between wheels in each axle is adjustable to fit different cross section profiles. The main chassis consists of two side members, each part holds 2 horizontal wheels and 2 vertical wheels. The vertical wheels guide the vehicle model and prevent it from wandering. Figure 6 shows the side members and the wheels assembly.

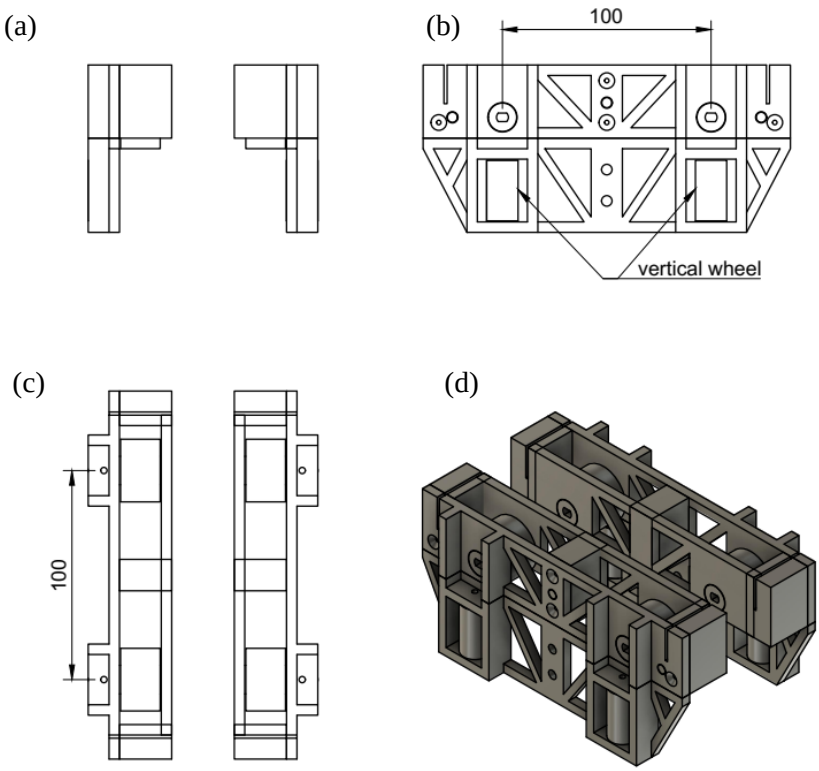


Figure 6 The main Chassis of the vehicle model: (a) front view, (b) left view, (c) top view, (d) 3D view.

The vehicle is equipped with two DC motors and gearboxes. The gearbox is used in order to provide sufficient torque to move the vehicle. The gear ratio can be adjusted to 5 different values depending

on the torque and speed required. As shown in figure 7, the gearbox consists of five double helical gears with different number of teeth.

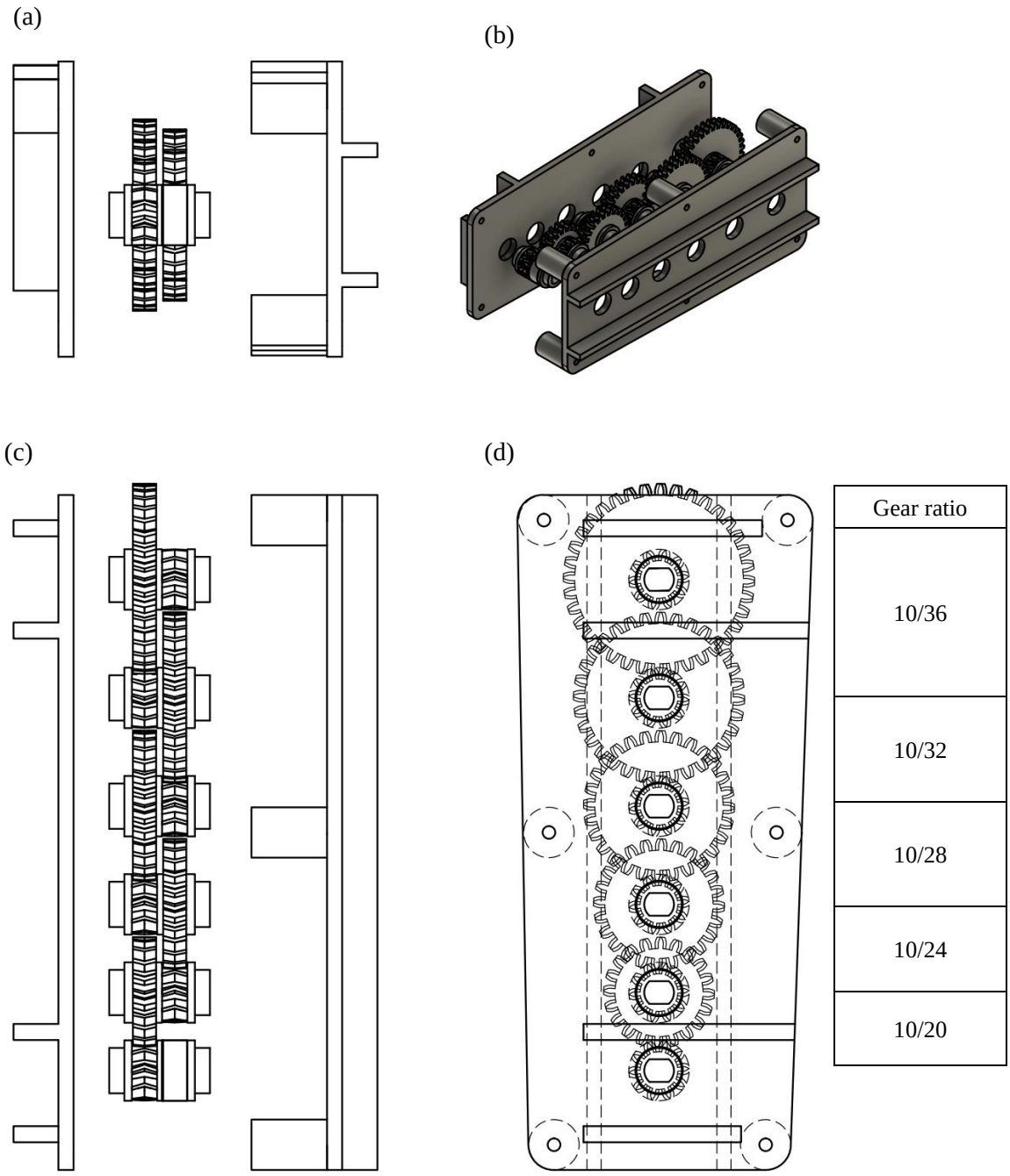


Figure 7 The gearbox components and gear ratio: (a) front view, (b) 3D view, (c) top view, (d) gear ratio stages.

The upper part was designed to support an extra weight and protect the other component. The components were printed using Ender 3D printer and the material used was PLA filament. The figures 8-9 demonstrate the different parts of the model.

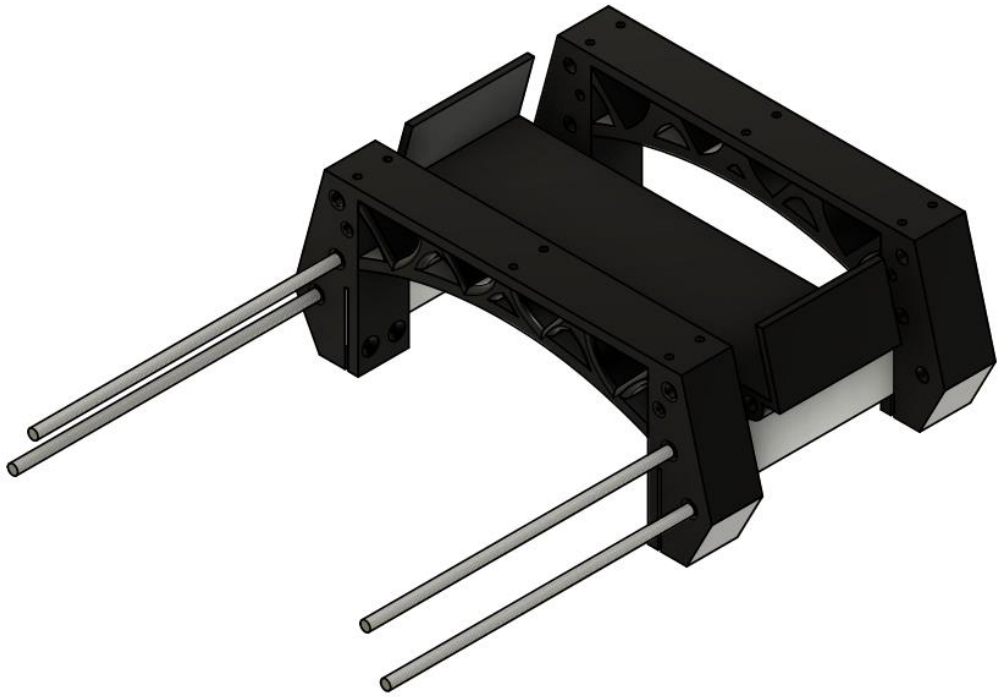


Figure 8 The top part of the vehicle.

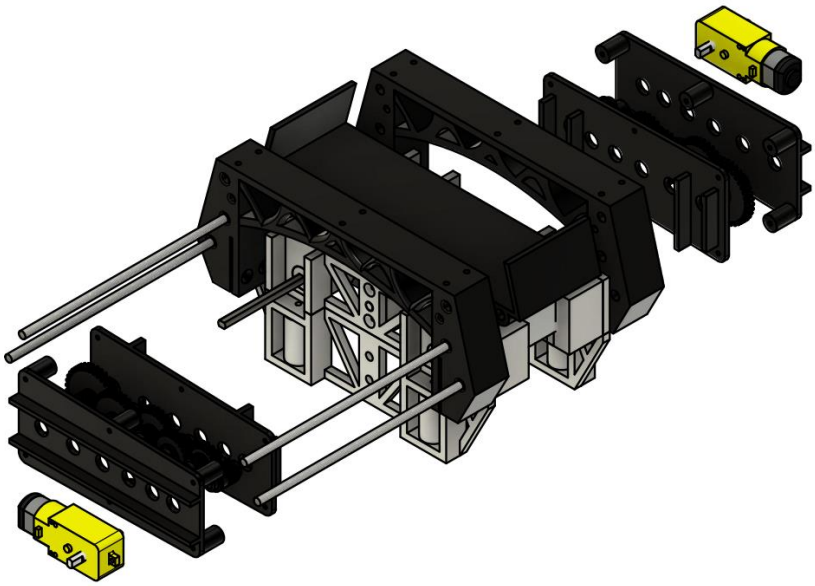


Figure 9 The mechanical components of the vehicle.

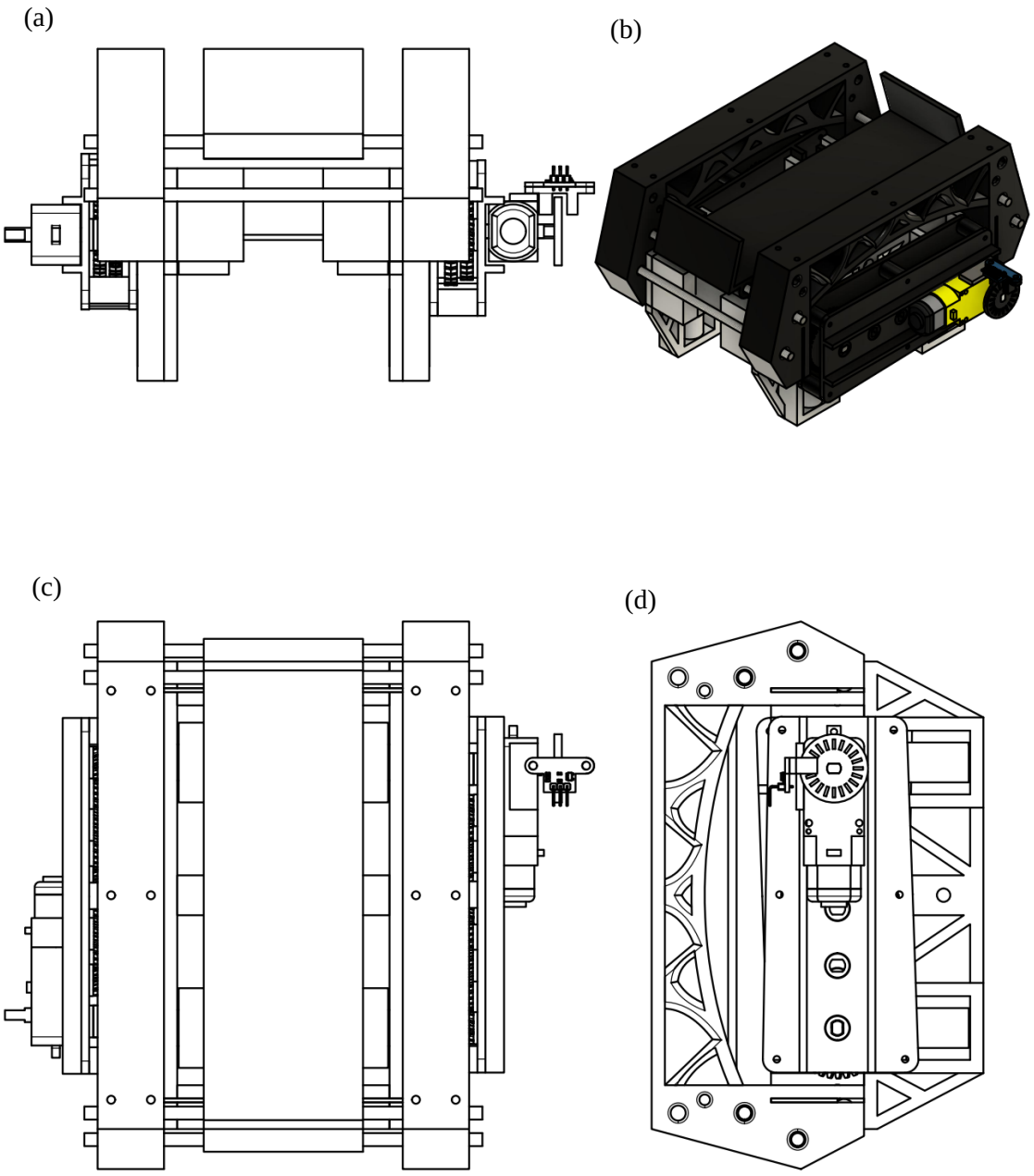


Figure 10 The final assembly of the mechanical parts: (a) front view, (b) 3D view, (c) top view, (d) left view.

Figures 11-12 demonstrate the printed model after the installation of all the mechanical parts and the hardware. The wheels were covered with synthetic rubber to prevent sliding and smoothen the movement of the vehicle.

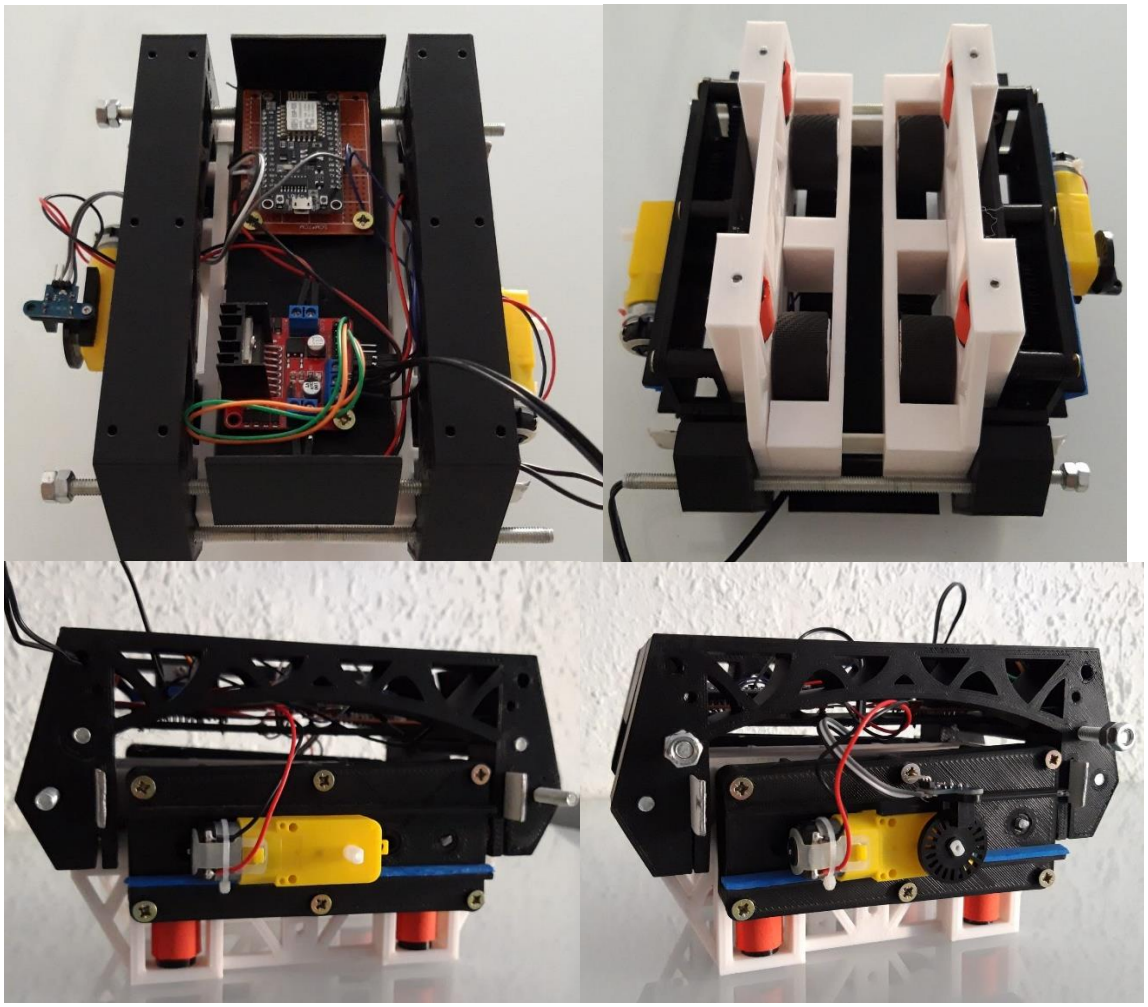


Figure 11 The printed parts assembled and fixed.

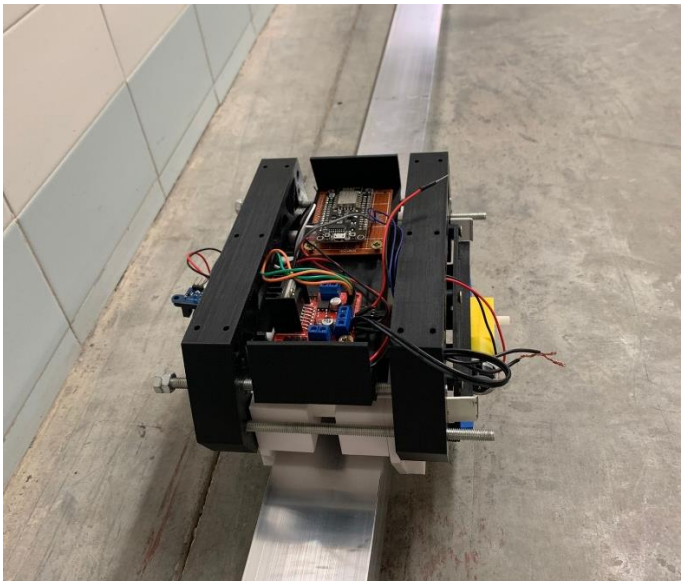


Figure 12 The vehicle model over beam of 40x60mm cross-section profile.

4.2. The hardware of the vehicle model

The motors are controlled using NodeMCU chipset, which contains an ESP8266 microcontroller with Wi-Fi capability. L298 dual H-Bridge driver was interfaced with NodeMCU to power the motors with enough voltage and current since NodeMCU cannot handle the motor heavy power load.

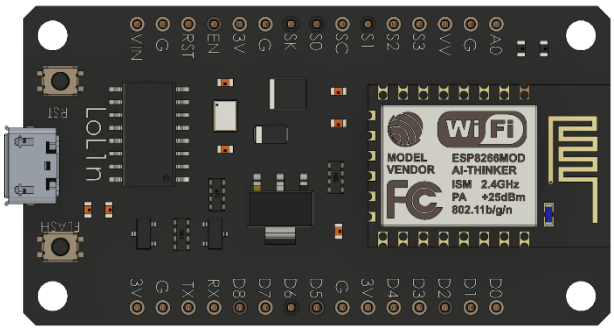


Figure 13 NodeMCU Lolin V3.

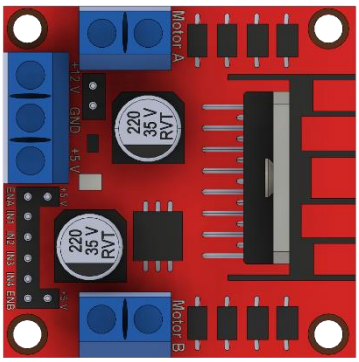


Figure 14 L298 dual H-Bridge driver.

To measure the speed of the vehicle an optical encoder was attached to one of the DC motors as shown in figure 15. The sensor signals then processed by NodeMCU microcontroller to control the motors speed. The optical encoder can also measure the distance travelled by the vehicle.

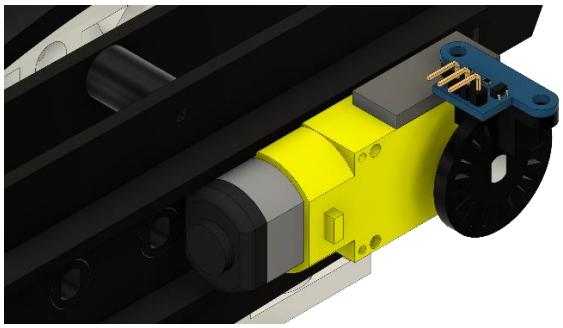


Figure 15 The optical encoder.

The hardware components were already assembled on PCBs and interfaced with each other as shown in figure 16. The pin D5 and D6 are connected to L298 circuts througn EN1 and EN2 pins respectively to enable the right motor and control its speed. EN1 and EN2 pin are linked with EN3 and EN4 pin respectively, as there is no need to dedicate another output pins on NodeMCU to control the left motor. Both motors are powered by the L298 circuit output. The optical encoder signal is transmited to NodeMCU through pin D2.

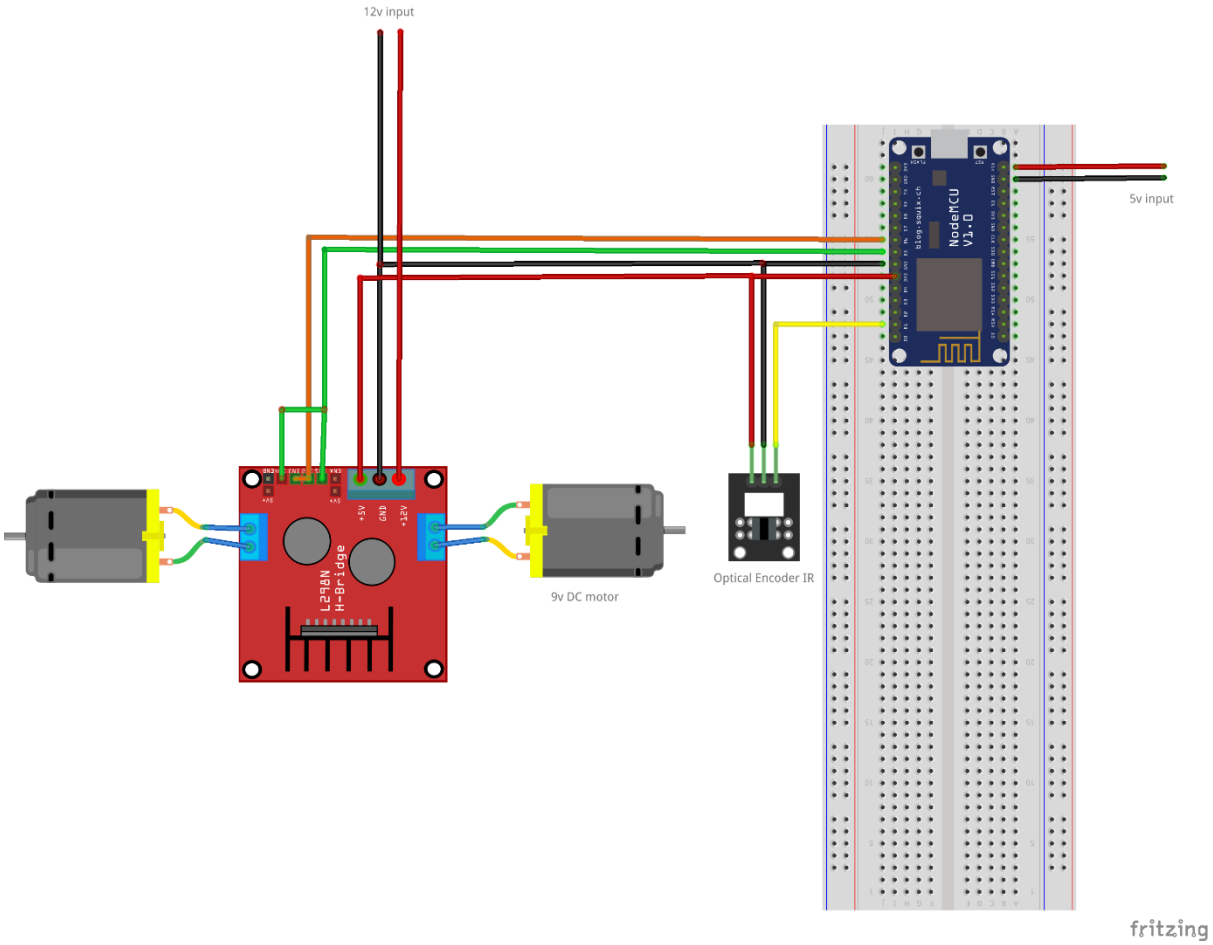


Figure 16 The schematic of hardware.

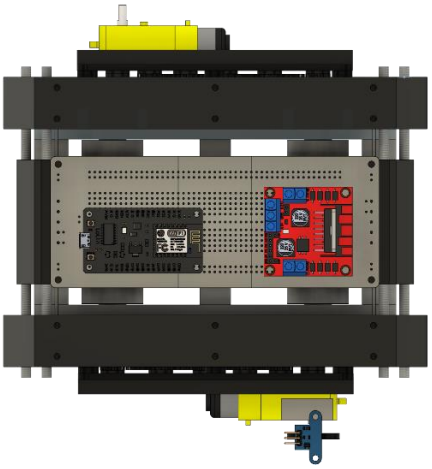


Figure 17 The hardware installation.

4.3. The software of the vehicle model

The software consists of 3 blocks of code, Arduino code for the NodeMCU, live server, and Python application with user interface. The three blocks are presented in appendix A with explanation.

The NodeMCU was programmed using Arduino platform and the code was written in C/C++. The code provides NodeMCU with the following functionalities:

- UDP server for online data transmission.
- Configures D5 and D6 pins as output with pulse wave modulation to adjust the power of the motors.
- Speed and distance calculation.

The NodeMCU is programed to establish connection with an online server through an available Wi-Fi network. The data transmitted online through the UDP server consists of 4 bytes instructions, first byte is the instruction code and the last three are the value of the instruction. The instruction received from the server is used to control the motors, and the data sent is the speed and the distance.

Using the optical encoder signal the distance and speed of the vehicle model can be calculated. The speed is then adjusted simultaneously to the required speed value by the NodeMCU.

The server was written in Python and deployed on Ubuntu Amazon Web server. The server enables the control application and the vehicle model to communicate through an API gateway. The server provides static IP address and listen on port 80.

The application used to control and configure the vehicle was written in Python. The application can be run on any operating system that has Python installed, and it establish connection with the server to communicate with the vehicle model. The vehicle speed and travelled distance can be configured and monitored through the application. The actual speed and distance are displayed simultaneously while the vehicle is moving. Figure 18 shows the application user interface and the main input and output.

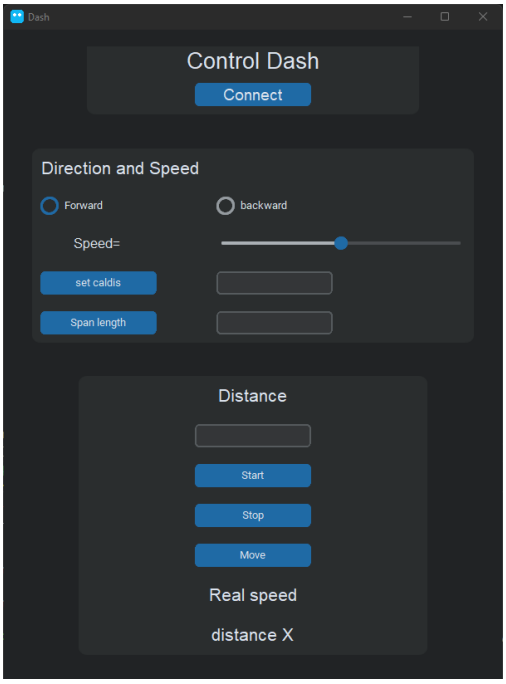


Figure 18 The user interface of the application.

5. Laboratory experiments

In this section, LARA's measurement accuracy is evaluated by comparing its results with the estimations of a HI-INC inclinometer in four tests. Then, the combinatory analysis presents the accuracy improvement of inclination measurement of up to five combined inclinometers. Finally, the accuracy and resolution of HI-INC and LARA are validated by performing four load tests on a simply supported aluminum beam.

5.1. Accuracy evaluation

In this section, the experimental tests targeted at verifying the accuracy of LARA are shown. In order to make sure that LARA and the commercial inclinometers measure the same inclination, LARA was glued on top of the HI-INC clinometer. Then, the HI-INC was connected to the rigid metallic plate using its magnetic plate. After that, the metallic plate was connected to a rotational device (Figure 19). By rolling the small gear of this rotational device, the connected stiff plate rotates. Then, induced rotations were measured by the LARA and HI-INC. Finally, the outputs of LARA for the tests were compared with those estimated by the HI-INC inclinometer.

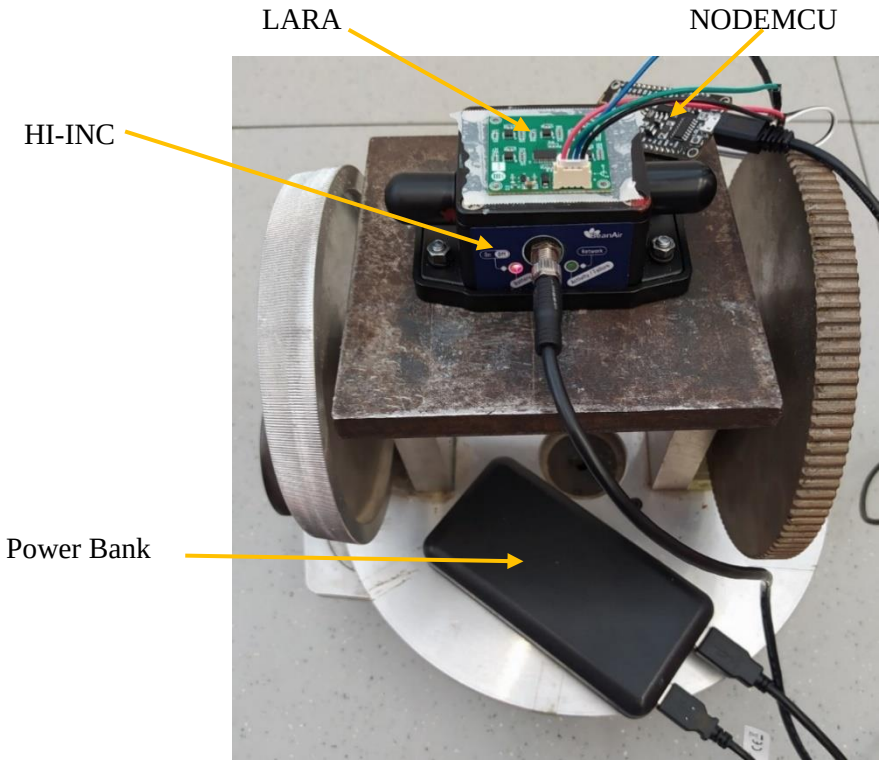


Figure 19 Test setup intended for comparing inclination estimation of LARA with HI-INC.

Table 2 Accuracy comparison of LARA with HI-INC

N	HI-INC (DEGREES)	LARA (DEGREES)	DIFFERENCE (DEGREES)
1	0.9996	0.9615	0.0382
2	1.9770	1.9267	0.0503
3	3.0180	2.9618	0.0563
4	4.0254	3.9583	0.0671

Table 2 presents the results of the carried out experimental tests. This table takes in the following information collected in columns: (1) N: Four tests are carried out for evaluating the accuracy of LARA in different inclinations, (2) HI-INC: the estimated inclination by HI-INC inclinometer, (3) LARA: the measured inclination of LARA, and (4) Difference: the absolute difference of LARA and HI-INC measurements.

The analysis of table 2 shows that the difference in LARA measurement from the HI-INC is related to the induced inclination. In fact, it was seen that for more than five degrees of change, the difference of LARA from HI-INC was higher than 0.1 degrees. For that reason, their data are not included in Table 2. Therefore, it can be concluded that the accurate measuring range of LARA is up to four degrees. This range is accurate enough for the target application of bridge monitoring as in this kind of structures increments of rotation higher than 0.5 degrees are not expected [23][77].

5.2. Combinatory analysis

In order to study the difference of the measured values from the reference sensor for different inclinometer combinations, a combinatory analysis was performed. This evaluation illustrates the maximum and minimum envelope difference from the commercial inclinometer for all the possible sensor selections from the five available inclinometers. The maximum and minimum values of the increasing number of inclinometers are shown in Figures 20, 20.a (one sensor), 20.b (two sensors), 20.c (three sensors), and 20.d (four sensors). It is to be noted that n all these figures LARA shows the estimation calculated by combining the results of five inclinometers together.

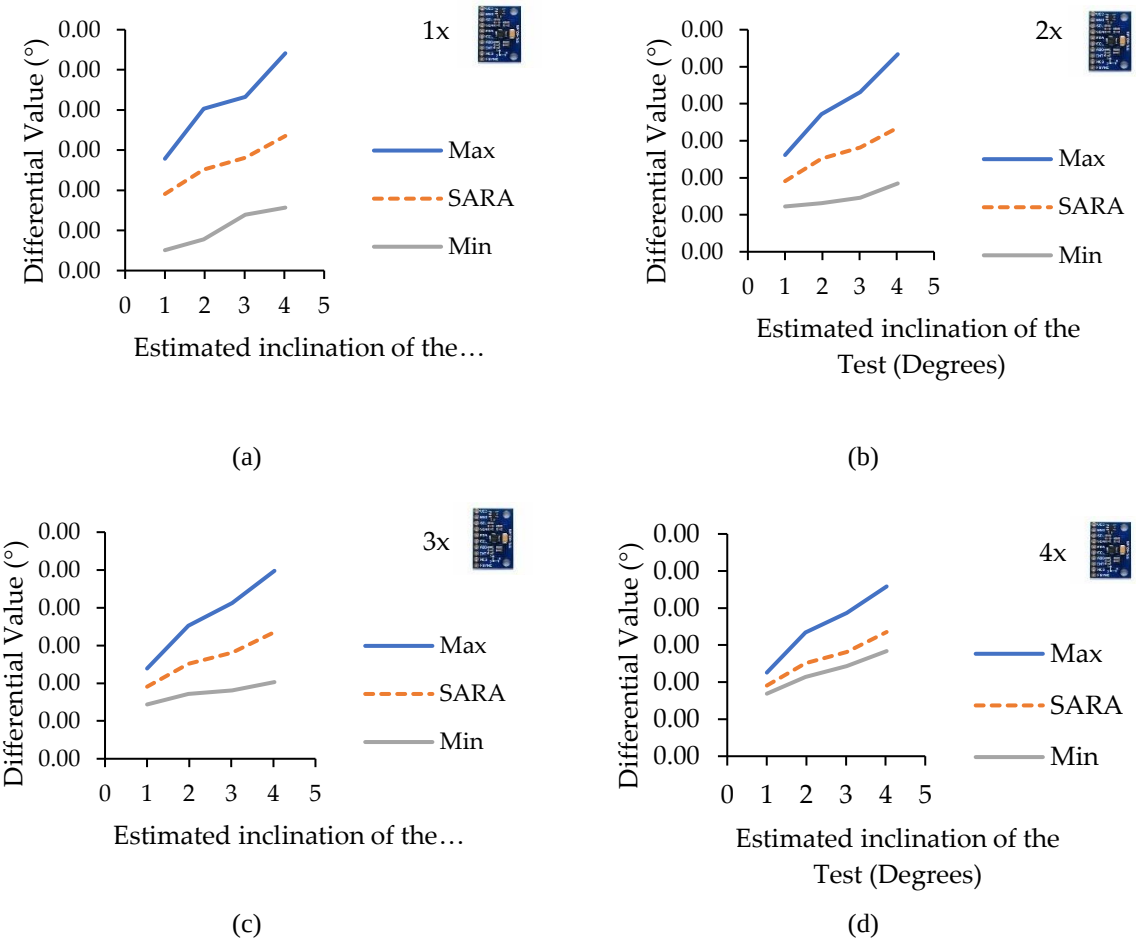


Figure 20 Estimated measured inclination difference for a different number of combined inclinometers from HI-INC estimations: One sensor (a), two sensors (b), three sensors (c), and four sensors (d).

The analysis of Figure 20 shows that the accuracy of the whole system is directly influenced by the number of combined inclinometers. For example, the minimum accuracy of a single inclinometer (Max difference from HI-INC) is 0.0557 degrees for an induced inclination of 0.9996. However, for the same experiment, LARA showed a measurement difference of 0.0381 from HI-INC. As expected, the higher the number of sensors, the better the accuracy of the modular system. It is essential to note that the minimum difference from HI-INC estimations reported in Figure 20.a does not correspond to the measurement of a single sensor for all four experiments. It can also be seen that the distance between minimum and the maximum differential from HI-INC values is decreasing with a higher number of combined sensors. In fact, having a lower range of possible errors can help making the final product more reliable. This reliability is very important when an inclinometer has a high sampling frequency. This way, optimizing filters (such as different Kalman filter formulations [78]), which could alter the primary signal and slow down the acquisition speed, are no longer necessary.

5.3. LARA resolution and accuracy verification using a beam model

In order to present the resolution and accuracy of LARA more clearly, a load test is performed on a small-scale beam with a length of 1.24m. This section compares the slope estimation of two sensors (LARA and H-INC) located on the support of a simply supported aluminum beam model under a point load of 467 gr (4.58 kN) with hand calculation of slope at the beam edges. It should be mentioned that, for this test, LARA was again mounted on the top of the HI-INC.

This test is carried out using a U-shaped aluminum profile with section dimensions of 25×25×3×3 mm. The effective length of the beam model, which is the distance between the null axis of its support, is fixed as 1080 mm.

The test aim was to read the maximum slope of the beam model deck under a known applied load on the mid-span. The maximum slope at the supports can be calculated by Equation 5. Therefore, LARA and HI-INC were attached to achieve this objective on top of the beam model support. First, LARA and HI-INC worked for a while without any loads (Figure 21.a) and their estimations were acquired. Next, the point load was set on the mid-span of the beam model (Figure 21.b and 21.c) and then another data acquisition process was carried out to measure the slope of the beam by LARA and HI-INC. It is essential to mention that this test was repeated three times.

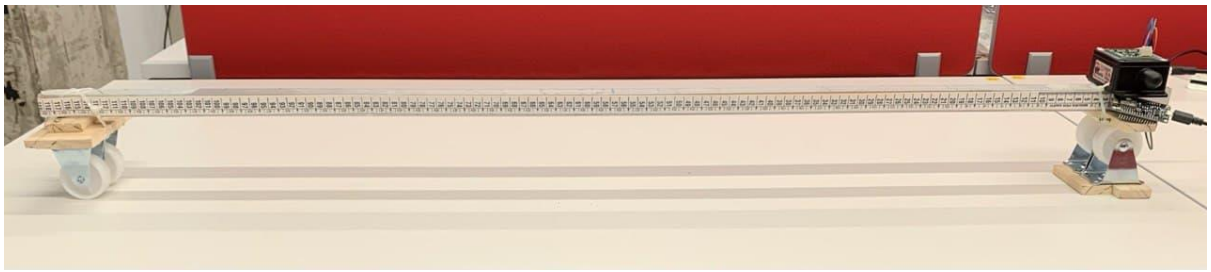
The used formula for calculating the slope of a simply supported beam with a load located on its midspan by hand is presented in Eq.5 [79].

$$\Delta\theta_1 = \frac{P \times L^2}{16 \times E \times I} \tag{5}$$

In Eq.5, where, $\Delta\theta_1$ (Radians) is the maximum slope at the supports, P is the value of the applied load at the mid-span, L is the effective beam length, E (69637.05 MPa) is the beam elasticity module, and I (12853.08 mm^4) is the beam moment of inertia. $\Delta\theta$ is then calculated as 0.000373 radians. This value corresponds to 0.021372 degrees on inclination. The comparison of the estimated values of LARA and HI-INC with those of the theoretical estimation is presented in Table 3.

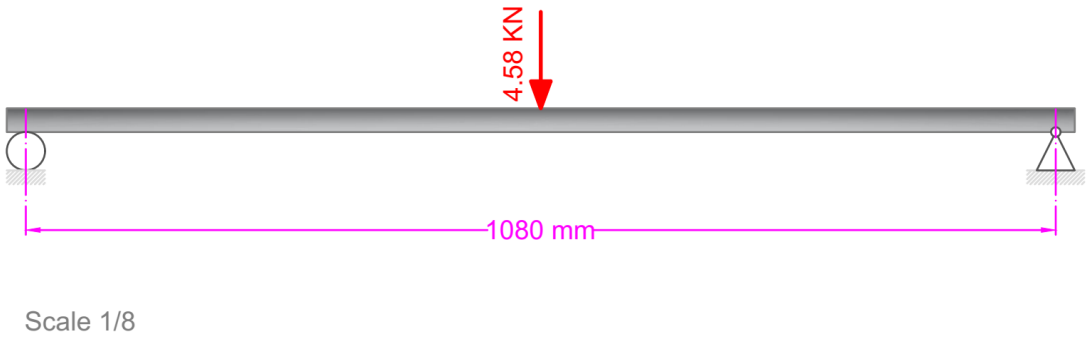
Table 3 Comparing the inclination estimation of LARA and HI-INC

Number of the experiments	Hand calculation slope (degrees)	LARA Difference (degrees)	LARA (degrees)	HI-INC difference (degrees)	HI-INC (degrees)
1	0.021372	0.001613	0.022985	0.002447	0.018925
2	0.021372	0.002316	0.023688	0.000853	0.020519
3	0.021372	0.001362	0.022734	0.005196	0.016176



(a)

(b)



1080 mm (Effective length)
(c)

Figure 21 Load test of a beam model: (a) test setup, (b) load test, and (c) sketch of the load test.

The analysis of Table 3 shows that the accuracy of LARA based on these experiments is less than 0.002 degrees. Further study of Table 3 illustrates that the accuracy of HI-INC is around 0.005 degrees. In fact, this is very close to accuracy value detailed in its datasheet (±0.003° for ±15° version) [80]. This value validates the accountability of the performed experiment. Therefore, having accuracy in the range of 0.05 degrees makes LARA applicable for the SHM of bridges.

Another experimental test was carried out on this beam model (Figure 21.a) using a heavier weight (21.942 N). In this experiment, instead of putting the weight only on the midspan, the weight was set on various beam locations. Then, the support slope was measured using HI-INC and LARA. Finally, the sensors' measurements are compared with the theoretical estimation [79]. Figure 22 presents the slope measurement comparison of HI-INC and LARA with the hand calculation values.

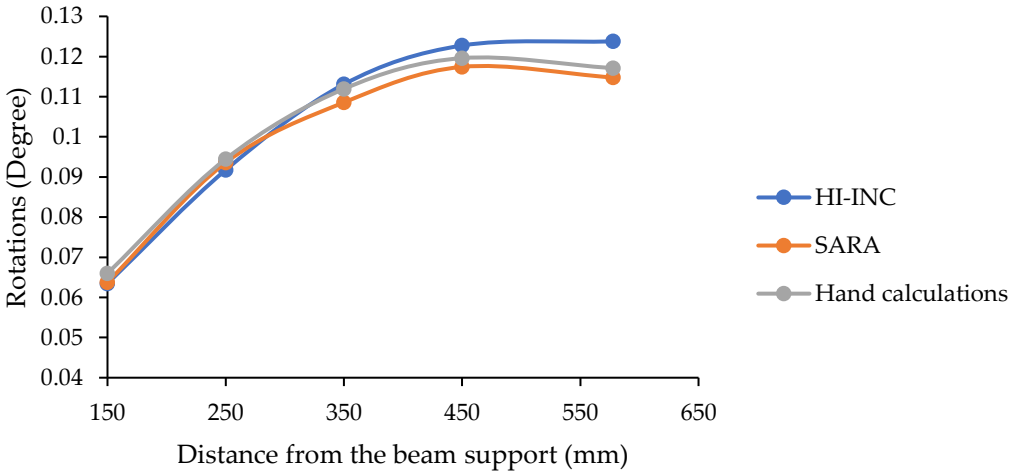


Figure 22 Support slope of a simply supported beam under a point load located on various spots

Analysis of Figure 22 shows that LARA has a maximum measured difference of 0.003 degrees from the hand calculation slope. In addition, it can be seen that LARA has a closer trend to the hand calculation values compared to those of HI-INC.

It should be noted that LARA can be used in static load tests aiming to identify the location of structural damages which had altered the influence line of a bridge [23].

It is interesting to compare LARA's final price (54.95 €) with those presented in Table 1. It should be noted that comparing an academically developed device with a commercial alternative is not fair. However, the most critical contribution of current work is developing a low-cost, accurate device and for that, this comparison is needed. It can be seen from Table 1 that HI-INC, ZCT-CX09 and DNS have a resolution of 0.003 degrees. Therefore, LARA can be compared with them. Figure 23 presents the price comparison of these inclinometers.

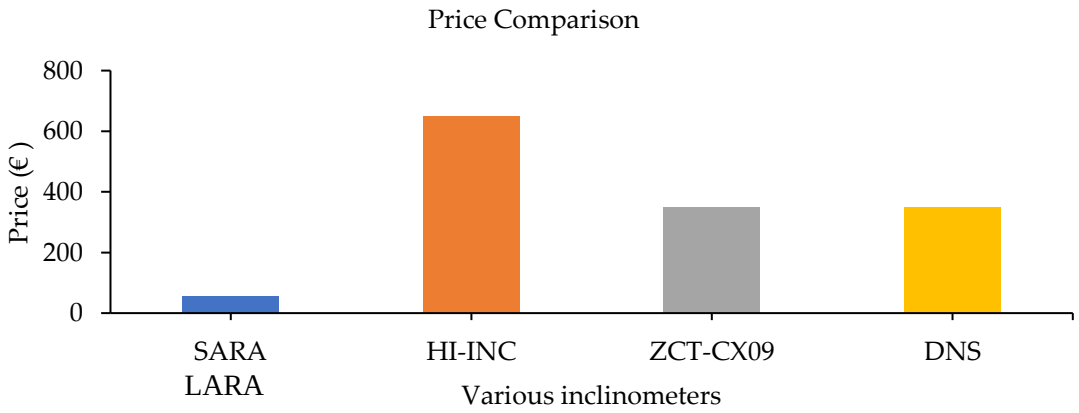


Figure 23 Price comparison of LARA with traditional commercial inclinometers with a resolution of 0.003 degrees.

Analysis of Figure 23 shows a significant difference between the price of LARA and inclinometers with the same resolution. LARA is 12, 6 and 6 times cheaper than HI-INC, ZTC-CX09 and DNS inclinometers, respectively. Also, it does not need extra paid commercial software for data acquisition and it is based on open-source software and hardware.

6. Conclusion

Recently, engineers and researchers have been paying a lot of attention to the inclinometer application for SHM of bridges. Inclinometers, as opposed to accelerometers, make it simple to assess the magnitude and location of structural damage. They are useful for long-term structural health monitoring (SHM) of bridges due to this property. Additionally, inclinometers may be used to quickly determine the deflection of a structural element. However, the expensive cost of today's inclinometers has restricted their application. The invention of a low-cost inclinometer for long-term SHM of bridges with a minimal budget for their health inspections leaves a gap in the literature.

This research introduces a Low-cost Adaptable Reliable Angle-meter (LARA) technology to fill these gaps. LARA is a low-cost, wireless, Internet of Things-based inclinometer with a sampling rate of 250 Hz. Five inclinometers (MPU9250), a multiplexor, and an Internet of Things-based microcontroller make up the device (NodeMCU). Every inclinometer uses a complimentary filter to merge the data it has collected from its accelerometer and gyroscope. LARA's key innovation is its ability to combine the data of five aligned inclinometers in order to lower the inherent noise density of its separate accelerometers and gyroscopes.

Four laboratory experiments were conducted to verify the concept of noise reduction and signal improvement of inclination measurements using the averaged outcomes of various aligned inclinometers. These tests' results demonstrated that averaging a number of aligned accelerometers' values lowers the noise density of the frequency domain representation of a vibration acquisition experiment. Additionally, it is demonstrated that a system made up of five aligned inclinometers has substantially lower Allan variance and deviation than a single inclinometer.

An experimental test was carried out to validate the measurement of LARA in a rotation range of zero to four degrees in order to validate the accuracy of LARA. In this experiment, the values of a commercial inclinometer were compared to the data collected by LARA. In tests with one and four degrees of inclination, respectively, LARA is observed to present up to 0.04 and 0.07 degrees of variation. The structure rarely anticipates experiencing a slope of more than 0.5 degrees in SHM of bridges.

A load test on a beam is also conducted to evaluate the accuracy of the LARA and the commercial inclinometer that is being used. In this test, LARA's results are contrasted with those of the commercial inclinometer. It is demonstrated that LARA's theoretical slope estimation differed from the manually obtained values by less than 0.003 degrees. However, HI-INC demonstrated precision with a magnitude of 0.005° that was higher than the data in its datasheet.

Another experiment was planned to be carried out on a 40x60x600 profile using the vehicle model as a moving load. The experiment implements direct rotation measurement to detect damage and its location using LARA inclinometer. However, due to the late delivery of the profile, we could not conduct the experiment and was left for future investigation.

LARA is relevant to bridge SHM and inclinometer-specific damage detection methods due to its high rotation estimation accuracy. However, LARA needs to be verified in a bridge load test application in further study.

7. Future research

Several topics have been brought to light through the research done for this thesis that need further research and development.

The literature review identified several gaps in the body of knowledge. Some of these were addressed through the research for this thesis, but others still need to be addressed. There aren't enough studies on using inclinometers in structural damage detection. Specifically, whether there is any potential for detecting damage in a complex bridge structure under real traffic loads using inclinometers. Future research may, for instance, search for patterns in the data harvested from the damage detection system and define a realistic damage index.

Additionally, there are many areas where the work done for this thesis might be applied and developed further. In this study, LARA has been introduced as an accurate and economical inclinometer. However, LARA can be integrated with different technologies such as weigh-in-motion and GPS to detect damage in real-life bridge cases and under real traffic loading. This would provide a real-time picture of the performance of the bridge as a whole and allow comparison of the outcomes when various damage detection methodologies are utilized.

This work also established another area for further development which is integration Robotics in the damage detection system. With the large access to cutting-edge technology, it is now possible to create devices and robots designed specifically to perform inspection and different tasks in the process of SHM. Robots and UAV can reach inaccessible parts of any structure, they can collect important data of the structure condition using their sensors and assist the damage detection system from installation to maintenance.

7.1. Future steps

The future research lines are summarized as follows:

1. First step is to address Damage detection using Inclinometers as a statistical pattern recognition problem, which can be solved following four steps: (1) Operational evaluation, (2) Data acquisition, (3) Feature extraction, and (4) Statistical modeling for feature classification. By using this strategy, we can obtain the damage index which provides a realistic performance evaluation and risk assessment.
2. The damage index will then be validated in laboratory by carrying out experiments and analyzing the collected data. The experiments would be automated and accelerated using robotics systems to reduce time consumption and errors due to human interaction. LARA with the vehicle model will be used to detect damage in a bridge model under different boundary conditions and loading scenarios. In this stage, LARA will also be validated under dynamic loading.
3. The damage detection system will be integrated with Weigh in Motion and GPS technology, which can provide the necessary information about the traffic loads in real time. This process is essential because without knowing the load conditions damage prognosis cannot be performed.
4. Developing a software framework to interface the components of the damage detection system. It will be necessary to provide reliable and efficient scripts and libraries for each element in the system. This will include the development of statistical model and damage prognosis algorithms where Machine learning and its other subset can be utilized to optimize the detection methodology.
5. The detection system can be tested on a real bridge for further improvement and validation. Data acquired from real cases can be used to improve the algorithms used in the statistical model and the damage prognosis which can include Neural Networks.
6. Robotics and UAV will be introduced in the damage detection system. Robots can be

designed to work as remote sensors which can reduce the number of sensors needed. Moreover, Robots and UAV can install the sensors in their places and reduce the risk of human injuries.

7.2. Related works and publications

The following papers is published in a Q2 journal (Sensors) as a consequence of this thesis:

1. Seyedmilad Komarizadehasl, Mahyad Komary, **Ahmad Alahmad**, Jose Antonio Lozano-Galant, Gonzalo Ramos and Jose Turmo, A Novel Wireless Low-Cost Inclinometer Made from Combining the Measurements of Multiple MEMS Gyroscopes and Accelerometers, Published at Sensors journal, July 2022. https://doi.org/10.3390/s22155605

The following papers are published in international conferences as consequences of this thesis:

2. Seyedmilad Komarizadehasl, **Ahmad Alahmad**, Víctor TORRALBA, Jose A. Lozano-Galant, and Jose Turmo, Beneficial Effect of Combining Similar Low-Cost Accelerometer to improve the overall Accuracy and Noise Density, IABSE Congress, Nanjing, 2022, September 21-23.
3. Seyedmilad Komarizadehasl, **Ahmad Alahmad**, Víctor TORRALBA, Jose A. Lozano-Galant, and Jose Turmo, Experimental Verification of A Novel Accelerometer Intended For Structural Health Monitoring of Bridges, IABSE Congress, Nanjing, 2022, September 21-23.

References

[1] M. R. Kaloop, M. Eldiasty, and J. W. Hu, "Safety and reliability evaluations of bridge behaviors under ambient truck loads through structural health monitoring and identification model approaches," *Measurement*, vol. 187, p. 110234, Jan. 2022, doi: 10.1016/J.MEASUREMENT.2021.110234.

[2] S. Komarizadehasl, B. Mobaraki, H. Ma, J.-A. Lozano-Galant, and J. Turmo, "Development of a Low-Cost System for the Accurate Measurement of Structural Vibrations," *Sensors*, vol. 21, no. 18, pp. 6191–6213, Sep. 2021, doi: 10.3390/S21186191.

[3] S. Komarizadehasl, B. Mobaraki, H. Ma, J.-A. Lozano-Galant, and J. Turmo, "Low-Cost Sensors Accuracy Study and Enhancement Strategy," *Applied Sciences 2022, Vol. 12, Page 3186*, vol. 12, no. 6, pp. 3186–3215, Mar. 2022, doi: 10.3390/APP12063186.

[4] D. Proske, "Fatalities due to bridge collapse," *Proceedings of the Institution of Civil Engineers - Bridge Engineering*, vol. 173, no. 4, pp. 1–24, Jun. 2020, doi: 10.1680/jbren.20.00001.

[5] J. Farré-Checa, S. Komarizadehasl, H. Ma, J. A. Lozano-Galant, and J. Turmo, "Direct simulation of the tensioning process of cable-stayed bridge cantilever construction," *Autom Constr*, vol. 137, p. 104197, May 2022, doi: 10.1016/J.AUTCON.2022.104197.

[6] D. Straub *et al.*, "Value of information: A roadmap to quantifying the benefit of structural health monitoring," in *12th International Conference on Structural Safety & Reliability*, 2017, no. 12, pp. 3018–3029.

[7] Z. Zhou, S. Shao, G. Deng, Y. Gao, S. Wang, and X. Chu, "Vision-based modal parameter identification for bridges using a novel holographic visual sensor," *Measurement*, vol. 179, p. 109551, Jul. 2021, doi: 10.1016/J.MEASUREMENT.2021.109551.

[8] B. R. K. Mantha, C. C. Menassa, and V. R. Kamat, "Robotic data collection and simulation for evaluation of building retrofit performance," *Autom Constr*, vol. 92, pp. 88–102, Aug. 2018, doi: 10.1016/j.autcon.2018.03.026.

[9] G. Fedorko, V. Molnár, M. Vasiľ, and R. Salai, "Proposal of digital twin for testing and measuring of transport belts for pipe conveyors within the concept Industry 4.0," *Measurement*, vol. 174, p. 108978, Apr. 2021, doi: 10.1016/J.MEASUREMENT.2021.108978.

[10] U. Vitiello, V. Ciotta, A. Salzano, D. Asprone, G. Manfredi, and E. Cosenza, "BIM-based approach for the cost-optimization of seismic retrofit strategies on existing buildings," *Autom Constr*, vol. 98, pp. 90–101, Feb. 2019, doi: 10.1016/j.autcon.2018.10.023.

[11] J. Lei, J. A. Lozano-Galant, D. Xu, and J. Turmo, "Structural system identification by measurement error-minimizing observability method," *Struct Control Health Monit*, vol. 26, no. 10, pp. e2425–e2444, Oct. 2019, doi: 10.1002/stc.2425.

[12] J. A. Lozano-Galant, M. Nogal, J. Turmo, and E. Castillo, "Selection of measurement sets in static structural identification of bridges using observability trees," *Computers and Concrete*, vol. 15, no. 5, pp. 771–794, May 2015, doi: 10.12989/cac.2015.15.5.771.

[13] A. Entezami, H. Sarmadi, B. Behkamal, and S. Mariani, "Health Monitoring of Large-Scale Civil Structures: An Approach Based on Data Partitioning and Classical Multidimensional Scaling," *Sensors*, vol. 21, no. 5, p. 1646, Feb. 2021, doi: 10.3390/s21051646.

[14] S. Komarizadehasl and M. Khanmohammadi, "Novel plastic hinge modification factors for damaged RC shear walls with bending performance," *Advances in Concrete Construction*, vol. 12, no. 4, pp. 355–365, 2021, doi: 10.12989/ACC.2021.12.4.355.

[15] A. Nokhbatolfoghahai, H. M. Navazi, and R. M. Groves, "Evaluation of the sparse reconstruction and the delay-and-sum damage imaging methods for structural health monitoring under different environmental and operational conditions," *Measurement*, vol. 169, p. 108495, Feb. 2021, doi: 10.1016/J.MEASUREMENT.2020.108495.

[16] V. Spitas, C. Spitas, and P. Michelis, "Real-time measurement of shear fatigue crack propagation at high-temperature using the potential drop technique," *Measurement*, vol. 41, no. 4, pp. 424–432, May 2008, doi: 10.1016/J.MEASUREMENT.2007.01.003.

[17] X. Jian, Y. Xia, J. A. Lozano-Galant, and L. Sun, "Traffic sensing methodology combining influence line theory and computer vision techniques for girder bridges," *J Sens*, vol. 2019, 2019, doi: 10.1155/2019/3409525.

[18] J. Ramli, J. Coulson, J. Martin, B. Nagaratnam, K. Pologanathan, and W. M. Cheung, "Crack Detection and Localisation in Steel-Fibre-Reinforced Self-Compacting Concrete Using Triaxial Accelerometers," *Sensors*, vol. 21, no. 6, pp. 2044–2064, Mar. 2021, doi: 10.3390/s21062044.

[19] L. Zhang, X. Cheng, G. Wu, and T. Wang, "Reference-free damage identification method for highway continuous girder bridges based on long-gauge fibre Bragg grating strain sensors," *Measurement*, vol. 195, p. 111064, May 2022, doi: 10.1016/J.MEASUREMENT.2022.111064.

[20] L. Lei, D. Song, Z. Liu, X. Xu, and Z. Zheng, "Displacement Identification by Computer Vision for Condition Monitoring of Rail Vehicle Bearings," *Sensors*, vol. 21, no. 6, pp. 2100–2119, Mar. 2021, doi: 10.3390/s21062100.

[21] M. Gaitan and J. Geist, "Calibration of triaxial accelerometers by constant rotation rate in the gravitational field," *Measurement*, vol. 189, p. 110528, Feb. 2022, doi: 10.1016/J.MEASUREMENT.2021.110528.

[22] P. Antunes, H. Varum, and P. André, "Uniaxial fiber Bragg grating accelerometer system with temperature and cross axis insensitivity," *Measurement*, vol. 44, no. 1, pp. 55–59, Jan. 2011, doi: 10.1016/J.MEASUREMENT.2010.09.013.

[23] D. Hester, J. Brownjohn, F. Huseynov, E. Obrien, A. Gonzalez, and M. Casero, "Identifying damage in a bridge by analysing rotation response to a moving load," *Structure and Infrastructure Engineering*, vol. 16, no. 7, pp. 1050–1065, Jul. 2019, doi: 10.1080/15732479.2019.1680710.

[24] N. K. Raghuwanshi and A. Parey, "Experimental measurement of mesh stiffness by laser displacement sensor technique," *Measurement*, vol. 128, pp. 63–70, Nov. 2018, doi: 10.1016/J.MEASUREMENT.2018.06.035.

[25] X. Yao, Z. Xing, Z. Zhang, and A. Sheng, "The online monitoring system of pantograph slider based on 2D laser displacement sensors," *Measurement*, vol. 194, p. 111083, May 2022, doi: 10.1016/J.MEASUREMENT.2022.111083.

[26] H. Park, S. Son, S. Choi, and Y. Kim, "Wireless laser range finder system for vertical displacement monitoring of mega-trusses during construction," *sensors*, vol. 13, pp. 5796–5813, 2013, doi: 10.3390/s130505796.

[27] Y. Li, H. Wang, W. Cai, S. Li, and Q. Zhang, "Stability monitoring of surrounding rock mass on a forked tunnel using both strain gauges and FBG sensors," *Measurement*, vol. 153, p. 107449, Mar. 2020, doi: 10.1016/J.MEASUREMENT.2019.107449.

[28] X. Iriarte, J. Aginaga, G. Gainza, J. Ros, and J. Bacaicoa, "Optimal strain-gauge placement for mechanical load estimation in circular cross-section shafts," *Measurement*, vol. 174, p. 108938, Apr. 2021, doi: 10.1016/J.MEASUREMENT.2020.108938.

[29] E. Copertaro, "Assessment of resistive strain gauges measurement performances in experimental modal analysis and their application to the diagnostics of abrasive waterjet cutting machinery," *Measurement*, vol. 188, p. 110626, Jan. 2022, doi: 10.1016/J.MEASUREMENT.2021.110626.

[30] M. Majumder, T. Gangopadhyay, A. Chakraborty, K. Dasgupta, and D. Bhattacharya, "Fibre Bragg gratings in structural health monitoring—Present status and applications," *Sens Actuators A Phys*, doi: https://doi.org/10.1016/j.sna.2008.04.008.

[31] F. Huseynov, C. Kim, E. J. OBrien, J. M. W. Brownjohn, D. Hester, and K. C. Chang, "Bridge damage detection using rotation measurements – Experimental validation," *Mech Syst Signal Process*, vol. 135, p. 106380, Jan. 2020, doi: 10.1016/J.YMSSP.2019.106380.

[32] D. W. Ha, H. S. Park, S. W. Choi, and Y. Kim, "A Wireless MEMS-Based Inclinometer Sensor Node for Structural Health Monitoring," *Sensors 2013, Vol. 13, Pages 16090-16104*, vol. 13, no. 12, pp. 16090–16104, Nov. 2013, doi: 10.3390/S131216090.

[33] B. Glišić, D. Posenato, D. Inaudi, and A. Figini, "Structural health monitoring method for curved concrete bridge box girders," *https://doi.org/10.1117/12.778643*, vol. 6932, pp. 39–47, Apr. 2008, doi: 10.1117/12.778643.

[34] Y. Zhou, Z. Dongjian, C. Zhuoyan, and L. Yongtao, "Research on a novel inclinometer based on distributed optical fiber strain and conjugate beam method," *Measurement*, vol. 153, p.

107404, Mar. 2020, doi: 10.1016/J.MEASUREMENT.2019.107404.

[35] S. Bas, N. M. Apaydin, A. Ilki, and F. N. Catbas, "Structural health monitoring system of the long-span bridges in Turkey," *https://doi.org/10.1080/15732479.2017.1360365*, vol. 14, no. 4, pp. 425–444, Apr. 2017, doi: 10.1080/15732479.2017.1360365.

[36] J. M. Ko and Y. Q. Ni, "Technology developments in structural health monitoring of large-scale bridges," *Eng Struct*, vol. 27, no. 12, pp. 1715–1725, Oct. 2005, doi: 10.1016/J.ENGSTRUCT.2005.02.021.

[37] H. Pei, F. Zhang, and S. Zhang, "Development of a novel Hall element inclinometer for slope displacement monitoring," *Measurement*, vol. 181, p. 109636, Aug. 2021, doi: 10.1016/J.MEASUREMENT.2021.109636.

[38] N. Haritos and T. J. Chalko, "Determination of abutment support conditions in an 80-year-old RC bridge," *https://doi.org/10.1117/12.259148*, vol. 2946, pp. 312–323, Nov. 1996, doi: 10.1117/12.259148.

[39] N. A. Hoult, P. R. A. Fidler, P. G. Hill, and C. R. Middleton, "Long-Term Wireless Structural Health Monitoring of the Ferriby Road Bridge," *Journal of Bridge Engineering*, vol. 15, no. 2, pp. 153–159, Feb. 2010, doi: 10.1061/(ASCE)BE.1943-5592.0000049.

[40] F. Stajano, N. Hoult, I. Wassell, P. Bennett, C. Middleton, and K. Soga, "Smart bridges, smart tunnels: Transforming wireless sensor networks from research prototypes into robust engineering infrastructure," *Ad Hoc Networks*, vol. 8, no. 8, pp. 872–888, Nov. 2010, doi: 10.1016/J.ADHOC.2010.04.002.

[41] X. Hou, X. Yang, and Q. Huang, "Using Inclinometers to Measure Bridge Deflection," *Journal of Bridge Engineering*, vol. 10, no. 5, pp. 564–569, Sep. 2005, doi: 10.1061/(ASCE)1084-0702(2005)10:5(564).

[42] H. F. A. J. J. Sousa, "Bridge deflection evaluation using strain and rotation measurements," *Smart Struct Syst*, vol. 11, no. 4, pp. 365–386, 2013, doi: 10.12989/SSS.2013.11.4.365.

[43] K. Helmi, T. Taylor, A. Zarafshan, and F. Ansari, "Reference free method for real time monitoring of bridge deflections," *Eng Struct*, vol. 103, pp. 116–124, Nov. 2015, doi: 10.1016/J.ENGSTRUCT.2015.09.002.

[44] Y. Robert-Nicoud, B. Raphael, O. Burdet, and I. F. C. Smith, "Model Identification of Bridges Using Measurement Data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 20, no. 2, pp. 118–131, Mar. 2005, doi: 10.1111/J.1467-8667.2005.00381.X.

[45] D. Erdenebat, D. Waldmann, F. Scherbaum, and N. Teferle, "The Deformation Area Difference (DAD) method for condition assessment of reinforced structures," *Eng Struct*, vol. 155, pp. 315–329, Jan. 2018, doi: 10.1016/J.ENGSTRUCT.2017.11.034.

[46] K. Alten, M. Ralbovsky, A. Vorwagner, H. Toplitzer, and S. Wittmann, "Evaluation of Different Monitoring Techniques during Damage Infliction on Structures," *Procedia Eng*, vol. 199, pp. 1840–1845, 2017, doi: 10.1016/J.PROENG.2017.09.106.

[47] "MEMS Gyroscope Provides Precision Inertial Sensing in Harsh, High Temperature Environments | Analog Devices." https://www.analog.com/en/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html (accessed Oct. 16, 2021).

[48] T. Hiller, Z. Pentek, J. T. Liewald, A. Buhmann, and H. Roth, "Origins and Mechanisms of Bias Instability Noise in a Three-Axis Mode-Matched MEMS Gyroscope," *Journal of Microelectromechanical Systems*, vol. 28, no. 4, pp. 586–596, Aug. 2019, doi: 10.1109/JMEMS.2019.2921607.

[49] S. Ghasemi-Moghadam and M. R. Homaeinezhad, "Attitude determination by combining arrays of MEMS accelerometers, gyros, and magnetometers via quaternion-based complementary filter," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 31, no. 3, p. e2282, May 2018, doi: 10.1002/JNM.2282.

[50] K. Jerath, S. Brennan, and C. Lagoa, "Bridging the gap between sensor noise modeling and sensor characterization," *Measurement*, vol. 116, pp. 350–366, Feb. 2018, doi: 10.1016/J.MEASUREMENT.2017.09.012.

[51] F. Mumuni and A. Mumuni, "Adaptive Kalman filter for MEMS IMU data fusion using enhanced covariance scaling," *Control Theory and Technology 2021 19:3*, vol. 19, no. 3, pp. 365–374, Aug. 2021, doi: 10.1007/S11768-021-00058-8.

[52] R. I. Alfian, A. Ma'Arif, and S. Sunardi, "Noise Reduction in the Accelerometer and Gyroscope Sensor with the Kalman Filter Algorithm," *Journal of Robotics and Control (JRC)*, vol. 2, no. 3, pp. 180–189, May 2021, doi: 10.18196/JRC.2375.

[53] G. Ruzza, L. Guerriero, P. Revellino, and F. M. Guadagno, "A Multi-Module Fixed Inclinometer for Continuous Monitoring of Landslides: Design, Development, and Laboratory Testing," *Sensors 2020, Vol. 20, Page 3318*, vol. 20, no. 11, p. 3318, Jun. 2020, doi: 10.3390/S20113318.

[54] S. Komarizadehasl, B. Mobaraki, J. A. Lozano-Galant, and J. Turmo, "Detailed evaluation of low-cost ranging sensors for structural health monitoring applications," in *International Conference of Recent Trends in Geotechnical and Geo-Environmental Engineering and Education. "RTCEE/RTGEE 2020*, 2020, pp. 8–12.

[55] Y. Yu, J. Ou, J. Zhang, C. Zhang, and L. Li, "Development of wireless MEMS inclination sensor system for swing monitoring of large-scale hook structures," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 4, pp. 1072–1078, 2009, doi: 10.1109/TIE.2009.2012469.

[56] B. Andò, S. Baglio, and A. Pistorio, "A low cost multi-sensor system for investigating the structural response of buildings," *Annals of Geophysics*, vol. 61, no. 2, p. SE217, Jun. 2018, doi: 10.4401/ag-7702.

[57] M. L. Hoang and A. Pietrosanto, "A Robust Orientation System for Inclinometer with Full-Redundancy in Heavy Industry," *IEEE Sens J*, vol. 21, no. 5, pp. 5853–5860, Mar. 2021, doi: 10.1109/JSEN.2020.3040374.

[58] S. M. Khan, N. Qaiser, and M. M. Hussain, "An inclinometer using movable electrode in a parallel plate capacitive structure," *AIP Adv*, vol. 9, no. 4, p. 045118, Apr. 2019, doi: 10.1063/1.5092146.

[59] D. W. Ha, J. M. Kim, Y. Kim, and H. S. Park, "Development and application of a wireless MEMS-based borehole inclinometer for automated measurement of ground movement," *Autom Constr*, vol. 87, pp. 49–59, Mar. 2018, doi: 10.1016/j.autcon.2017.12.011.

[60] B. S. Chowdhry, A. A. Shah, M. A. Uqaili, and T. Memon, "Development of IOT Based Smart Instrumentation for the Real Time Structural Health Monitoring," *Wireless Personal Communications 2020 113:3*, vol. 113, no. 3, pp. 1641–1649, Apr. 2020, doi: 10.1007/S11277-020-07311-4.

[61] "Placa NodeMCU v3 - ESP8266 / CH340G Wifi BricoGeek CH340-V3 | BricoGeek.com." https://tienda.bricogeek.com/wifi/1033-nodemcu-v3-wifi-esp8266-ch340.html?search_query=nodemcu&results=5 (accessed Dec. 06, 2021).

[62] "Arduino Due - Modelo A000062 Arduino A000062 | BricoGeek.com." https://tienda.bricogeek.com/arduino-original/476-arduino-due.html?search_query=arduino+due&results=9 (accessed Dec. 06, 2021).

[63] F. Zonzini, A. Carbone, F. Romano, M. Zauli, and L. De Marchi, "Machine Learning Meets Compressed Sensing in Vibration-Based Monitoring," *Sensors 2022, Vol. 22, Page 2229*, vol. 22, no. 6, p. 2229, Mar. 2022, doi: 10.3390/S22062229.

[64] "Spectral density | TI.com Video." https://training.ti.com/ti-precision-labs-op-amps-noise-spectral-density (accessed Jun. 20, 2022).

[65] "BeanDevice® Wilow® User Manual." https://www.wireless-iot.beanair.com/files/UM-RF-07-ENG-Wilow-Wifi-Sensor.pdf (accessed Nov. 02, 2021).

[66] A. Knörig, R. Wettach, and J. Cohen, "Fritzing - A tool for advancing electronic prototyping for designers," in *3rd International Conference on Tangible and Embedded Interaction*, 2009, pp. 351–358. doi: 10.1145/1517664.1517735.

[67] G. Chiesa, D. Di Vita, A. Ghadirzadeh, A. H. Muñoz Herrera, and J. C. Leon Rodriguez, "A fuzzy-logic IoT lighting and shading control system for smart buildings," *Autom Constr*, vol. 120, p. 103397, Dec. 2020, doi: 10.1016/J.AUTCON.2020.103397.

[68] C. Yi *et al.*, "Estimating Three-Dimensional Body Orientation Based on an Improved Complementary Filter for Human Motion Tracking," *Sensors 2018, Vol. 18, Page 3765*, vol. 18, no. 11, p. 3765, Nov. 2018, doi: 10.3390/S18113765.

[69] X. Shen, M. Yao, W. Jia, and D. Yuan, "Adaptive complementary filter using fuzzy logic and simultaneous perturbation stochastic approximation algorithm," *Measurement*, vol. 45, no. 5,

pp. 1257–1265, Jun. 2012, doi: 10.1016/J.MEASUREMENT.2012.01.011.

[70] "HW VSP3 - Virtual Serial Port | HW-group.com." https://www.hw-group.com/software/hw-vsp3-virtual-serial-port (accessed Oct. 31, 2021).

[71] "SerialPlot - Realtime Plotting Software | Hackaday.io." https://hackaday.io/project/5334-serialplot-realtime-plotting-software (accessed Oct. 31, 2021).

[72] S. Komarizadehasl, S. Yuan, V. Torralba, J.-A. Lozano-Galant, and J. Turmo, "Low-Cost Wireless Structural Health Monitoring of Bridges," *Autom Constr*.

[73] N. K. *et al.*, "Noise modeling and analysis of an IMU-based attitude sensor: improvement of performance by filtering and sensor fusion," *https://doi.org/10.1117/12.2234255*, vol. 9912, pp. 2138–2147, Jul. 2016, doi: 10.1117/12.2234255.

[74] K.-M. Kwong, "MEMS Accelerometer Specifications and Their Impact in Inertial Applications," *Master thesis*. http://www.eecg.utoronto.ca/~johns/nobots/theses/pdf/2017_keiming_kwong_masc.pdf (accessed Mar. 07, 2021).

[75] K. Kwong, "MEMS Accelerometer Specifications and Their Impact in Inertial Applications," p. 95, 2017.

[76] J. Hidalgo, P. Poulakis, J. Köhler, J. Del-Cerro, and A. Barrientos, "Improving Planetary Rover Attitude Estimation via MEMS Sensor Characterization," *Sensors 2012, Vol. 12, Pages 2219-2235*, vol. 12, no. 2, pp. 2219–2235, Feb. 2012, doi: 10.3390/S120202219.

[77] C. McGeown, F. Huseynov, D. Hester, P. McGetrick, E. J. Obrien, and V. Pakrashi, "Using measured rotation on a beam to detect changes in its structural condition," *https://doi-org.recursos.biblioteca.upc.edu/10.1080/24705314.2021.1906092*, vol. 6, no. 3, pp. 159–166, 2021, doi: 10.1080/24705314.2021.1906092.

[78] M. Caruso *et al.*, "Analysis of the Accuracy of Ten Algorithms for Orientation Estimation Using Inertial and Magnetic Sensing under Optimal Conditions: One Size Does Not Fit All," *Sensors 2021, Vol. 21, Page 2543*, vol. 21, no. 7, p. 2543, Apr. 2021, doi: 10.3390/S21072543.

[79] R. C. Hibbeler, *Structural Analysis*. Pearson; 10th edition (July 28, 2017), ISBN: 0134610679, 2017.

[80] "Wireless Industrial IOT Inclinometer Sensor | Overview - BeanAir | Wireless IOT Sensors." https://www.beanair.com/wireless-iot-inclinometer-sensor-overview.html (accessed Sep. 17, 2021).

Appendix A

1. NodeMCU code

```
#include <Wire.h>                    //Include essential library
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

#ifndef UDP_TX_PACKET_MAX_SIZE
#define UDP_TX_PACKET_MAX_SIZE 4    // variable stores the maximum number of bytes
received per package
#endif

unsigned int localPort = 80;                      // constant stores local port
value the UDP server listen to
const char* remote_ip = "18.197.152.30";          // constant stores the remote
server IP
unsigned int remote_port = 80;                    // constant stores the remote
server port
const char* ssid     = "STASSID";                 // constant stores Wi-Fi SSID
const char* password = "STAPSK";                  // constant stores Wi-Fi password
// Variables for timing function
int T1 = 0;                                       // previous time value
int interval = 500;                               // interval time value
int ledState = LOW;                               // variable stores the built-in
led state

// Definition of the interruption function which handle the optical encoder signal
// This type of function simulates multiprocessing, it can handle input in real-
time without delay
const byte interruptPin = D2;                 // define the interrupt input pin
volatile int interruptCounter = 0;            // variable of volatile type to stores
the number of interuptions
void ICACHE_RAM_ATTR handleInterrupt();       // declaration of the interruption
function

// definitions and declarations of variables used in the loop function
char inst;                                    // variable stores instructions code
float ival = 0;                               // variable stores instructions value
char m_dir;                                   // variable stores direction of
movement
float m_speed = 0;                            // variable stores the required speed
of the motors
float b_speed = 25;                           // variable stores the initial speed
of the motors
float m_distance = 0;                         // variable stores the required
distance
```

```
float pdis = 0;                                  // variable stores the pervious
distance measured
volatile float dis = 0;                          // variable stores the distance
calculated using the optical encoder
char buf[10];                                    // variable stores temporary values
char spd[24];                                    // variable stores the actual speed
instruction to be sent to the server
char adis[24];                                   // variable stores the actual distance
instruction to be sent to the server
volatile float v = 0;                            // variable stores calibration value
of the optical encoder
                                                 // in order to calculate the distance
// The UDP server
WiFiUDP Udp;                 // declaration a class of type UDP server
// buffers for receiving and sending data
char packetBuffer[UDP_TX_PACKET_MAX_SIZE + 1]; // buffer to hold incoming packet,
the byte in the end indicate the end of the package.
char ReplyBuffer[] = "acknowledged\r\n";       // optional acknowledgment message

void setup() {
  Serial.begin(115200);   // configure the serial port bandwidth
  pinMode(D5, OUTPUT);    // output for motors
  pinMode(D6, OUTPUT);    // output for motors
  pinMode(LED_BUILTIN, OUTPUT); // enable the built-in led
  analogWriteFreq(20);          // configure the output PWM frequency

  pinMode(interruptPin, INPUT);         // input of the optical encoder signal
  pinMode(interruptPin, INPUT_PULLUP);  // keep the pin high
  attachInterrupt(interruptPin, handleInterrupt, FALLING);  // attach the
interruption function to the pin and define the trigger

  Wire.begin();                                  // initialize the serial
communication
  Serial.println();                              // print empty line on the serial
monitor

  WiFi.mode(WIFI_STA);                           // configure Wi-Fi mode to be
station
  WiFi.begin(ssid, password);                    // initialize the Wi-Fi
communication
  while (WiFi.status() != WL_CONNECTED) {        // connect to the Wi-Fi network
    delay(500);                                  // The while loop keep running
forever until the connection established
    Serial.print(".");
  }
  // Once the Wi-Fi connection established the code continue
  Serial.println("");                             // print empty line on the serial
monitor
```

```
  Serial.println("WiFi connected");
  Serial.println("IP address: ");                 // print the device local IP
  Serial.println(WiFi.localIP());

  Udp.begin(localPort);                           // initialize the UDP server
listening to port 80
  Serial.printf("UDP server on port %d\n", localPort);
  Serial.println();


  while (!Udp.parsePacket()) {                    // connect to the online server
    Udp.beginPacket(remote_ip, remote_port);      // The while loop keep running
forever until receiving response from the server
    Udp.write("device");
    Udp.endPacket();
    delay(1000);
    Serial.println(packetBuffer);
  }
}

// The interruption function executed once the trigger occurs without delay and
outside the main loop
// Every time the input of the interruption pin D2 goes from high to low the
trigger occurs
void handleInterrupt() {                          // define the interruption
function to handle the input from the optical encoder
  interruptCounter++;                             // counter of the number of
interruptions
  if (m_distance){                                // calculating distance. The
number of interruptions * the distance travelled during one interruption
  dis = interruptCounter * v;
  }

  if (dis >= m_distance &&m_distance ) {          // stop the motor once the
distance required is achived
    StopMotor();
    dtostrf(dis, 6, 2, buf);                      // prepare the distance
instruction to be sent to the server
    strcat(adis, "d");
    strcat(adis, buf);
    Udp.beginPacket(remote_ip, remote_port);      // begin transmission through UPD
server
    Udp.write(adis);                              // send the packet
    Udp.endPacket();                              // end packet
    memset(adis, '\0', sizeof(adis));             // clear the distance instruction
memory
    m_distance = 0;                               // reset the required distance
```

```
    interruptCounter = 0;                          // reset the number of
interruptions
  }
}

void loop() {                                      // The main loop runs forever
  if (millis() - T1 > interval)                    // Time function executed at
specific interval amount of time
  {
    T1 += interval;
    if ((dis - pdis) * 1000 / interval != 0) {          // the speed is calculated
by taking the difference of distance for one interval over the interval value
      if ((dis - pdis) * 1000 / interval < m_speed) {   // adjusting initial speed
to match the required speed
        b_speed += 25;
      }
    }

    pdis = dis;                                          // set the pervious
distance measured to the current one

    if (ledState == LOW)                                 // blink built-in led
in each interval
    {
      ledState = HIGH;
    }
    else
    {
      ledState = LOW;
    }
    digitalWrite(LED_BUILTIN, ledState);

    dtostrf((dis - pdis) * 1000 / interval, 6, 2, buf);     // prepare the speed
instruction to be sent to the server
    strcat(spd, "s");
    strcat(spd, buf);
    Udp.beginPacket(remote_ip, remote_port);             // begin transmission
through UPD server
    Udp.write(spd);                                      // send the packet
    Udp.endPacket();                                     // end packet
    memset(spd, '\0', sizeof(spd));                      // clear the speed
instruction memory
  }

  int packetSize = Udp.parsePacket();
  if (packetSize) {                                      // check if there is
packet received
```

```
    int n = Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE); // read the packet
into packetBufffer
    packetBuffer[n] = 0;                                    // assign zero to the
last byte as the end of the packet

    inst = packetBuffer[0];                                 // store the first
byte of the packet as the instruction code
    ival = String(packetBuffer).substring(1).toFloat();     // store the rest of
the packet as the instruction value
    // Processing the instructions and assign their value to the corresponding
variable
    if (inst == 'f') {                  // forward

      m_dir = inst;
      m_speed = ival;
    } else if (inst == 'b') {           // backward
      m_dir = inst;
      m_speed = ival;
    } else if (inst == 's') {           // stop
      m_dir = inst;
    } else if (inst == 'd') {           // set distance
      m_distance = ival;
    } else if (inst == 'c') {           // set calibration distance
      v = ival;
    }
  }

  // Enabling and control motors
  if (m_distance != 0) {                // check the required distance is not zero
    if (m_dir == 'f') {                 // if direction is "f" (forward), run
Forward function assigning the initial speed as argument
      Forward(b_speed);
    } else if (m_dir == 'b') {          // if direction is "b" (backward), run
Backward function assigning the initial speed as argument
      Backward(b_speed);
    } else if (m_dir == 's') {          // if direction is "s" (stop), set
required distance to zero
      m_distance = 0;
      dis = 0;
      interruptCounter = 0;
    }
  } else {                              // if the required distance is zero, run
StopMotor function
    StopMotor();
    b_speed = 25;                       // reset initial speed to its default
value
  }
```

```
}

// Definition of movement function to control the speed and direction of the
motors
void Forward(int speedarg)
{
  analogWrite(D5, speedarg);
  digitalWrite(D6, LOW);
  analogWrite(D7, speedarg);
  digitalWrite(D8, LOW);
}
void Backward(int speedarg)
{
  digitalWrite(D5, LOW);
  analogWrite(D6, speedarg);
  digitalWrite(D7, LOW);
  analogWrite(D8, speedarg);
}
void StopMotor()
{
  digitalWrite(D5, LOW);
  digitalWrite(D6, LOW);
  digitalWrite(D7, LOW);
  digitalWrite(D8, LOW);
}
```

2. The application code:

```
import threading                    #import the required libraries to run the
code
import socket
import tkinter
from tkinter import *
import customtkinter

localIP     = ""                    # IP variable declaration of the
application UDP server
localPort   = 80                    # port variable declaration of the
application UDP server
bufferSize  = 1024                  # variable declaration of the size of the
buffer to receive UDP packets
message=''                          # variable declaration of a message to be
sent
port=80                             # variable declaration of the remote
server port (server port)
address=('18.197.152.30',port)      # variable declaration of the remote
server address (remote server IP, remote server port)
s=""                                # variable declaration of a UDP client
```

```
dir_v ="f"                          # variable declaration of the vehicle
direction with initial value
speed_v ='0'                        # variable declaration of the vehicle
speed with initial value
v=0                                 # variable declaration of the received
vehicle speed with initial value
d=0                                 # variable declaration of the received
vehicle distance with initial value
BL=0                                # variable declaration of the length of
the beam

def main():                         # definition of the main function
    global s                        # include global variables
    global d                        #
    global message                  #
    global address                  #
    # Custom tkinter class definition
    customtkinter.set_appearance_mode("System")     # appearance configuration
of tkinter
    customtkinter.set_default_color_theme("blue")   # appearance configuration
of tkinter
    root = customtkinter.CTk()                      # window class definition
    root.geometry("600x900")                        # window class
configuration
    root.iconbitmap('botim.ico')                    # window class
configuration
    var = IntVar()                                  #
    root.title("Dash")                              # window class
configuration

    # UDP server function definition
    def startserver():                              #
        global v                                    # include global variables
        global dir_v
        global s
        global d
        global message
        global address
        if (Run.state == NORMAL):                   # check if the server is
already running
            Run.configure(state=tkinter.DISABLED)   # disable the connect
button if the server is running
            Run.configure(text="Connected")
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)    # create
socket class of UDP type
        s.bind((localIP, localPort))                            # assign the
local machine IP and port to the socket
```

```
        s.sendto("client".encode(), address)        # initialize connection
with the remote server
        # send defult configration values
        s.sendto(("f0").encode(), address)          # send initial
direction/speed instruction
        s.sendto("c0.443".encode(), address)        # send initial distance
calibration
        print("Socket successfully created")        #
        print("UDP server up and listening")        #
        while True:             # listen to the remote server
            bytesAddressPair = s.recvfrom(bufferSize)   # receive packets from
the remote server
            message = bytesAddressPair[0]               # extract the message
content of the packet
            # process the message content
            if message.decode("utf-8")[0]=='s':         # check if the
received message is the speed
                v=float(message.decode("utf-8")[1:7])   #
                if v>=0:
                    speedL.configure(text="Real speed: {}
mm/s".format(round(v)))   # display the speed value

            if message.decode("utf-8")[0]=='d':         # check if the
received message is the speed
                if dir_v=='f':                          # check the direction
                    d+=float(message.decode("utf-8")[1:7])  # calculate the
final distance
                elif dir_v=='b':                        # check the direction
                    d-=float(message.decode("utf-8")[1:7])  #calculate the
final distance
                distanceL.configure(text="X: {} mm".format(round(d,2))) #
display the final distance value

    def changedirection():      # definition of the direction function
        global dir_v
        global speed_v
        if var.get()==0:
            s.sendto(("f"+str(speed_v)).encode(), address)
            dir_v="f"
        else:
            s.sendto(("b"+str(speed_v)).encode(), address)
            dir_v="b"
    def changespeed(value):     # definition of the speed function
        global dir_v
        global speed_v
        if (address):
            s.sendto((dir_v+str(value)).encode(), address)
            aspeedL.configure(text="Speed: {} mm/s".format(str(value)))
```

```
            speed_v=str(value)
    def setcd(val):     # definition of the calibration distance function
        if (val==''):
            window = customtkinter.CTkToplevel(root)
            window.title("Warning!")
            window.geometry("300x200")
            window.iconbitmap('botimerror.ico')
            labelw = customtkinter.CTkLabel(window, text="Insert correct
value")
            labelw.pack(side="top", fill="both", expand=True, padx=40,
pady=40)
        else:
            s.sendto(("c"+str(val)).encode(), address)
    def setBL(val):     # definition of the beam length function
        global BL
        if (val==''):
            window = customtkinter.CTkToplevel(root)
            window.title("Warning!")
            window.geometry("300x200")
            window.iconbitmap('botimerror.ico')
            labelw = customtkinter.CTkLabel(window, text="Insert correct
value")
            labelw.pack(side="top", fill="both", expand=True, padx=40,
pady=40)
        else:
            BL=float(val)
    def setd(val):      # definition of the required distance function
        print(val)
        global dir_v
        x=float(val)
        if (d+x>BL and dir_v=='f' or d-x<0 and dir_v=='b'):
            pass
        else:
            s.sendto(("d"+str(val)).encode(), address)
    def stopm():        # definition of the motor disabling function
        if (address):
            s.sendto("s".encode(), address)
    def move():     # definition of distance alignment function
        global d
        s.sendto(("d"+str(0.5)).encode(), address)
        d=0

    # Define user interface elements of the main widow and assign their
functions
    frame0 = customtkinter.CTkFrame(master=root,
width=400,height=200,corner_radius=10)
    dashL = customtkinter.CTkLabel(frame0, text="Control
Dash",width=400,text_font=("CordiaUPC", 20))
```

```
    t1 = threading.Thread(target=startserver, name='t1', daemon = True)
    Run = customtkinter.CTkButton(frame0,
text="Connect",text_font=("CordiaUPC", 14), command=t1.start)
    frame =
customtkinter.CTkFrame(master=root,width=300,height=200,corner_radius=10)
    frameL = customtkinter.CTkLabel(frame, text="Direction and
Speed",text_font=("CordiaUPC", 16))
    forward_butt = customtkinter.CTkRadioButton ( frame,text="Forward",
variable=var, value=0, command=changedirection)
    backward_butt = customtkinter.CTkRadioButton ( frame,text="backward",
variable=var, value=1, command=changedirection)
    aspeedL = customtkinter.CTkLabel(frame, text="Speed=
",text_font=("CordiaUPC", 12))
    speedS = customtkinter.CTkSlider(frame, from_=20,
to=50,number_of_steps=6,width=300, orient=HORIZONTAL,command=changespeed)
    cdis = customtkinter.CTkEntry(frame)
    cdis_butt = customtkinter.CTkButton(frame, text="set caldis",
command=lambda: setcd(cdis.get()))
    frame1 = customtkinter.CTkFrame(master=root,
width=400,height=200,corner_radius=10)
    frame1L = customtkinter.CTkLabel(frame1,
text="Distance",width=400,text_font=("CordiaUPC", 16))
    distance = customtkinter.CTkEntry(frame1)
    distance_butt = customtkinter.CTkButton(frame1, text="Start",
command=lambda: setd(distance.get()))
    BLe = customtkinter.CTkEntry(frame)
    BL_butt = customtkinter.CTkButton(frame, text="Span length",
command=lambda: setBL(BLe.get()))
    stop_butt = customtkinter.CTkButton(frame1, text="Stop", command=stopm)
    move_butt = customtkinter.CTkButton(frame1, text="Move", command=move)
    speedL = customtkinter.CTkLabel(frame1, text="Real
speed",width=400,text_font=("CordiaUPC", 16))
    distanceL = customtkinter.CTkLabel(frame1, text="distance
X",width=400,text_font=("CordiaUPC", 16))

    # Place the UI elements on the main window
    frame0.pack(padx=20, pady=20)
    dashL.pack()
    Run.pack(padx=20, pady=10)
    frame.columnconfigure(0, weight=1)
    frame.columnconfigure(1, weight=2)
    frame.pack(padx=20, pady=20)
    frameL.grid(column=0, row=0, sticky="w", padx=10, pady=10)
    forward_butt.grid(column=0, row=1, sticky="w", padx=10, pady=10)
    backward_butt.grid(column=1, row=1, sticky="w", padx=10, pady=10)
    aspeedL.grid(column=0, row=2, sticky="w", padx=10, pady=10)
    speedS.grid(column=1, row=2, sticky="w", padx=10, pady=10)
    cdis.grid(column=1, row=3, sticky="w", padx=10, pady=10)
```

```
    cdis_butt.grid(column=0, row=3, sticky="w", padx=10, pady=10)
    BLe.grid(column=1, row=4, sticky="w", padx=10, pady=10)
    BL_butt.grid(column=0, row=4, sticky="w", padx=10, pady=10)
    frame1.pack(padx=20, pady=20)
    frame1L.pack(padx=10, pady=10,anchor="w")
    distance.pack(padx=20, pady=10)
    distance_butt.pack(padx=20, pady=10)
    stop_butt.pack(padx=20, pady=10)
    move_butt.pack(padx=20, pady=10)
    speedL.pack(padx=10, pady=10,anchor="w")
    distanceL.pack(padx=10, pady=10,anchor="w")

    # Initialize the main window
    root.mainloop()

# Make the code run standalone
if __name__ == '__main__':
    main()                  # start executing the main function
```

3. The server code:

```
from pickle import TRUE     #import the required libraries to run the code
from socket import *

bufferSize  = 1024          # variable declaration of the size of the buffer
to recevie UDP packets
def createServer():         # UDP server function definition
    serversocket = socket(AF_INET, SOCK_DGRAM)      # create socket class of
UDP type
    try :
        serversocket.bind(('172.31.33.6',80))       # assign the local machine
IP and port to the socket
        print("Socket successfully created")
        print("UDP server up and listening")
        while TRUE:         # start listening to income packet
            global clientID         # define IP client variable for the
application client
            global clientMsg        # define variable stores packets form the
application client
            global deviceID         # define IP device variable for the device
client
            global deviceMsg        # define variable stores packets form the
device client

            bytesAddressPair = serversocket.recvfrom(bufferSize) # store the
received packet in a buffer
            message = bytesAddressPair[0]                        # process
the buffer and extract the client address and the message content
```

```
            address = bytesAddressPair[1]
            # Define clients (application client, Device client)
            if message.decode("utf-8")=="device":               # If the
initializing message is "device", the IP of the device is stored
                deviceID  = address
                deviceMsg = message
                serversocket.sendto("ak".encode(), deviceID)    # acknowledge
connection with the device
                print("Device IP Address:{}".format(address))
                print("Message from Device:{}".format(message))
            elif message.decode("utf-8") =="client":            # If the
initializing message is "client", the IP of the application client is stored
                clientID  = address
                clientMsg = message
                serversocket.sendto("ak".encode(), clientID)    # acknowledge
connection with the application client
                print("Client IP Address:{}".format(address))
                print("Message from Client:{}".format(message))
            # forward messages between the device and the application client
            try:                        # check if there is a connected device
and client
                if address[1]==clientID:
                    serversocket.sendto(message, deviceID)
                else:
                    serversocket.sendto(message, clientID)
            except:
                print("well, No device connected after all!")

    except KeyboardInterrupt :                  # handling the server
termination
        print("\nShutting down...\n");
    except Exception as exc :
        print("Error:\n");
        print(exc)

createServer() # run the server function
```