VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF ELECTRICAL ENGINEERING

DINH NGUYEN GIA PHU

DESIGN OF DYNAMIC WEIGHING SYSTEM

INDIVIDUAL PROJECT

AUTOMATICAL AND CONTROL ENGINEERING

HO CHI MINH CITY, YEAR 2023

VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF ELECTRICAL ENGINEERING

DINH NGUYEN GIA PHU – 420H0265

DESIGN OF DYNAMIC WEIGHING SYSTEM

INDIVIDUAL PROJECT

AUTOMATICAL AND CONTROL ENGINEERING

Advised by

Mr. Thieu Quang Tri

HO CHI MINH CITY, YEAR 2023

ACKNOWLEDGMENT

No successful work is achieved without one's effort and dedication, as well as the help and support of teachers, friends, and colleagues around them. Therefore, I need to thank many wonderful people for their dedication and help in this project.

I would like to send my sincere thanks and good wishes to the esteemed teachers at Ton Duc Thang University who have devotedly instructed us during the past period. I thank the teachers in the Department of Electrical and Electronics, the Automation Department, and especially Mr. Thieu Quang Tri for guiding me in completing this individual project and helping me to gain more knowledge. This project would not have been completed without the help of the teachers. Due to my lack of experience and understanding, there may be mistakes that I hope the teachers will overlook and provide advice and guidance for me to draw experiences and complete the upcoming graduation project. I sincerely thank you all.

Finally, I would like to thank my parents and family for always accompanying and supporting me during the project.

Ho Chi Minh City, day month year

Author

Dinh Nguyen Gia Phu

DECLARATION OF AUTHORSHIP

I hereby declare that this thesis was carried out by myself under the guidance and supervision of Mr. Thieu Quang Tri; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews, and data used by other authors, and organizations have been acknowledged, and explicitly cited.

I will take full responsibility for any fraud detected in my thesis. Ton Duc Thang University is unrelated to any copyright infringement caused by my work (if any).

Ho Chi Minh City, day month year

Author

Dinh Nguyen Gia Phu

TEACHER'S CONFIRMATION AND ASSESSMENT SECTION

The confirmation part of the instructor

Ho Chi Minh City, *23 November 2023*

(Sign and write your full name)

The evaluation part of the teacher marks the test

Ho Chi Minh City, *23 November 2023*

(Sign and write your full name)

TON DUC THANG UNIVERSITY

FACULTY OF ELECTRICAL AND ELECTRONICS

SOCIALIST REPUBLIC OF VIETNAM

Independence - Freedom - Happiness

SCHEDULE OF INDIVIDUAL PROJECT

Student's name: Dinh Nguyen Gia Phu

Class: 20H40301 Student's ID: 420H0265

Topic: Design of dynamic weighting system.

Week	**Content**	**Instructor signed**
1	Research an overview of the topic and draw a block diagram.	
2	Learn about the components used in the project and draw an algorithm flow chart.	
3	Buy the components listed in the article and simulation on Proteus.	
4	Write code on software and test on test broad.	
5	Report 50%.	
6	Draw the PCB layout and iron up the circuit.	
7	Solder the components onto the circuit and test the results.	
8	Test the operating model before completing the system.	
9	Complete the model and write the report.	
10	Report 100%.	

TABLE OF CONTENT

Table of Contents

LIST OF FIGURE

LIST OF TABLE

LIST OF ABBREVIATIONS

LCD	Liquid Crystal Display
PWM	Pulse Width Modulation
I/O	Input/Output
DC	Direct Current
EMF	Electromotive Force
ADC	Analog-to-Digital Converter

CHAPTER 1. INTRODUCTION

1.1 OVERALL

Precision weighing and product speed monitoring on conveyor systems are critical for quality assurance and efficiency in modern manufacturing. However, traditional static weighing scales cannot provide measurements of weight for products moving on conveyors. Dynamic weighing systems allow for accurate weight measurement of products while they are moving down the conveyor line without stopping the belt.

The topic of this piece is about creating a dynamic weighing system with an Arduino microcontroller and components such as a load cell, encoder, LCD, and motor driver. The system is built around a conveyor belt that is powered by a 24VDC motor and whose speed is controlled by an L298N module. A 400 pulse per revolution encoder provides A and B channels to the Arduino to provide speed feedback. To measure the weight of objects moving over the sensor, a load cell mounted beneath the conveyor belt provides connectivity to the Arduino.

The Arduino Uno acquires both the load cell weight data and encoder pulse counts to calculate the real-time weight and speed of products on the moving conveyor belt. The measured speed and weight are shown on an LCD display for monitoring by an operator. Additionally, push buttons allow the operator to select between four preset conveyor speeds to adjust the weighing parameters or production flow rate.

This automated dynamic weighing system allows you to get essential product weight and conveyor speed data without interrupting the continuous flow of materials. The information supplied in this report will demonstrate the whole design, component selection, system integration, and testing techniques required for efficiently constructing the dynamic weighing system.

1.2 Reason for choosing the topic

This dynamic weighing system was created to answer a prevalent demand in many manufacturing and packaging plants. When motion is stopped, traditional static scales used for quality control and shipping confirmation only provide weight data. This causes an obstacle, limiting production line throughput and efficiency. The dynamic weighing system enables continuous weighing at full line speeds by allowing direct weight readings of things going on a conveyor. This capacity has the potential to provide various advantages to producers, including increased production, improved real-time quality control, reduced product waste, and automated data collecting for analysis.

The unique connection between a load cell, variable speed motor control, and encoder feedback allows for the development of specific solutions for the individual conveyor layouts utilized in various facilities. The capabilities established in this project will serve as a foundation for putting dynamic weighing on existing and future conveyor systems to improve industrial operations' efficiency and analytics. The Arduino platform enables the relatively straightforward integration of the essential electrical and mechanical components to offer a low-cost dynamic weighing system.

1.3 Target implementation

The dynamic weighing system described in this study is intended for easy integration into an extensive amount of industrial conveyor applications. The ability to accurately measure product weight on the move can benefit many manufacturing and distribution processes that currently rely on intermittent static weighing. Some major target implementation areas include:

- Food processing lines for precise portioning and quality control.
- Packaging facilities for carton filling and confirming shipped weights.
- Parcel sorting centers to automatically measure package weights during routing.
- Pharmaceutical manufacturing to monitor powder dispensing and tablet production.
- Postal and commercial shipping warehouses for automated weighing during sorting operations.
- Assembly and work cell lines to confirm subassembly or component weights.

The dynamic weighing system may be adjusted and configured to meet different belt widths and load ranges as needed due to its modular design that uses readily available components. The integrated motor speed control allows for operation at a variety of conveyor speeds. The dynamic weighing functionality adds automation, precision, and data collecting to important production and distribution processes while ensuring conveyor workflow continuity and throughput.

1.4 Object and scope of the study

The purpose of this project is to create and test a working dynamic weighing system design that can accurately measure object weight and conveyor speed for items going over the unit. Selecting appropriate load cell and encoder components, wiring them with an Arduino microcontroller, programming weight, and speed calculation logic, operating a variable speed motor, and displaying the output parameters are the main technical tasks. The project's scope includes the following elements:

- Researching and selecting suitable load cells, encoders, motors, and other components for integration into the system.
- Writing Arduino code to acquire, process, and display the load cell measurement and encoder pulse data.
- Integrating button controls for operator selection of motor speeds.
- Testing and calibration of the weighing system for accuracy under static and dynamic conditions.

1.5 Research method

This project will employ the following research methods:

- Literature Review: Technical papers, textbooks, and data sheets will be used to obtain background information on design concerns for load cells, encoders, motors, and other dynamic weighing system components. This will indicate viable options for future investigation.

- Component Selection: Based on the major criteria outlined in the literature review, particular components will be chosen by comparing specifications, pricing, and availability. Data sheets will be thoroughly examined.

- Experimentation: The working model will be tested using static and dynamic weights at various conveyor speeds. Data will be gathered to assess accuracy, precision, and repeatability.

- Analysis: To measure system performance and sensitivity, the experimental data will be analyzed statistically. The origins of errors and restrictions will be discovered.

- Refinement: Future manufacturing iterations will aim to increase performance through hardware and software optimizations driven by testing data analysis.

This combination of literature review, prototyping, experimentation, and analysis provides a disciplined technique for guiding the development of the dynamic weighing system from concept to prototype.

1.6 Practical significance

The creation of a functional dynamic weighing system model has several of significant practical consequences. Most importantly, it proves it's possible to precisely weigh goods as they travel down conveyor lines, rather than interrupting conveyor action. This functionality improves efficiency, throughput, and data collecting in a variety of manufacturing, packaging, and distribution settings that utilize conveyor activities. By reducing static weighing stoppages, facilities can achieve higher production rates and lower personnel expenses.

Overall, the functioning prototype represents a significant advancement in the use of dynamic weighing to improve efficiency, quality control, and data communication in materials handling processes.

1.7 Expected results

- Interfacing a load cell with an Arduino to acquire weight data.
- Using encoder pulses to calculate conveyor velocity.
- Implementing variable speed motor control through an L298N driver.
- Displaying weight and speed values on an LCD module.
- Meeting target accuracy and precision metrics based on testing.

In addition to the functional prototype, the project will have generated detailed documentation on the design, integration, programming, testing, and performance of the dynamic weighing system. Overall, the project aims to prove the feasibility of low-cost dynamic weighing using common electrical and mechanical components readily available to most technical teams.

CHAPTER 2. THEORETICAL BASIS

2.1 Overview of the Arduino Uno – Controller board

2.1.1 Introduction to Arduino Uno

The Arduino Uno is a extensively known 8-bit microcontroller board that is constructed based on the ATmega328P microprocessor. It is a adjustable Arduino family board that is considerably used for projects and prototypes in subject matter such as robotics, IoT, and embedded systems.

Through its digital I/O pins, the Arduino Uno supports a diversity of communication protocols such as I2C, SPI, and UART, making it capable of interacting with numerous sensors, actuators, and modules to develop particular applications.

The board also contains analog input pins, PWM output pins, hardware serial ports, and other onboard peripherals for connecting to external components and devices.

To program the Arduino Uno, the Arduino IDE, which uses a C/C++ based language framework, is used, making it simple even for beginners. The Arduino Uno comes with a plethora of libraries and code examples.

Overall, the Arduino Uno is a packed with functions, adaptable, and user-friendly microcontroller board that is suitable for hobbyists, students, and professionals looking to build electronics projects and prototypes in a range of fields.

2.1.2 Characteristics of Arduino Uno

The Arduino Uno microcontroller board has its foundation on the ATmega328P microprocessor. It is one of the most common boards in the Arduino family, and it is a fantastic choice for people who are new to microcontrollers.

Some key features of the Arduino Uno:

Feature	Description
Microcontroller	ATmega328P - 8-bit AVR microcontroller
Operating Voltage	5V
Recommended operating voltage	6 to 9V
Digital I/O Pins	14 (6 of which can be used as PWM outputs)
Analog Input Pins	6

DC Current per I/O Pin	20 mA
Clock Speed	16 MHz
USB Connection	For Programming
Maximum output current (5V)	500 mA
Maximum output current (3.3V)	50 mA
EEPROM	1KB
SRAM	2KB
Flash memory	32KB
Power Supply	Via USB or external power supply

Table 2-1 Specifications of Arduino Uno R3

2.1.3 Function of the ports

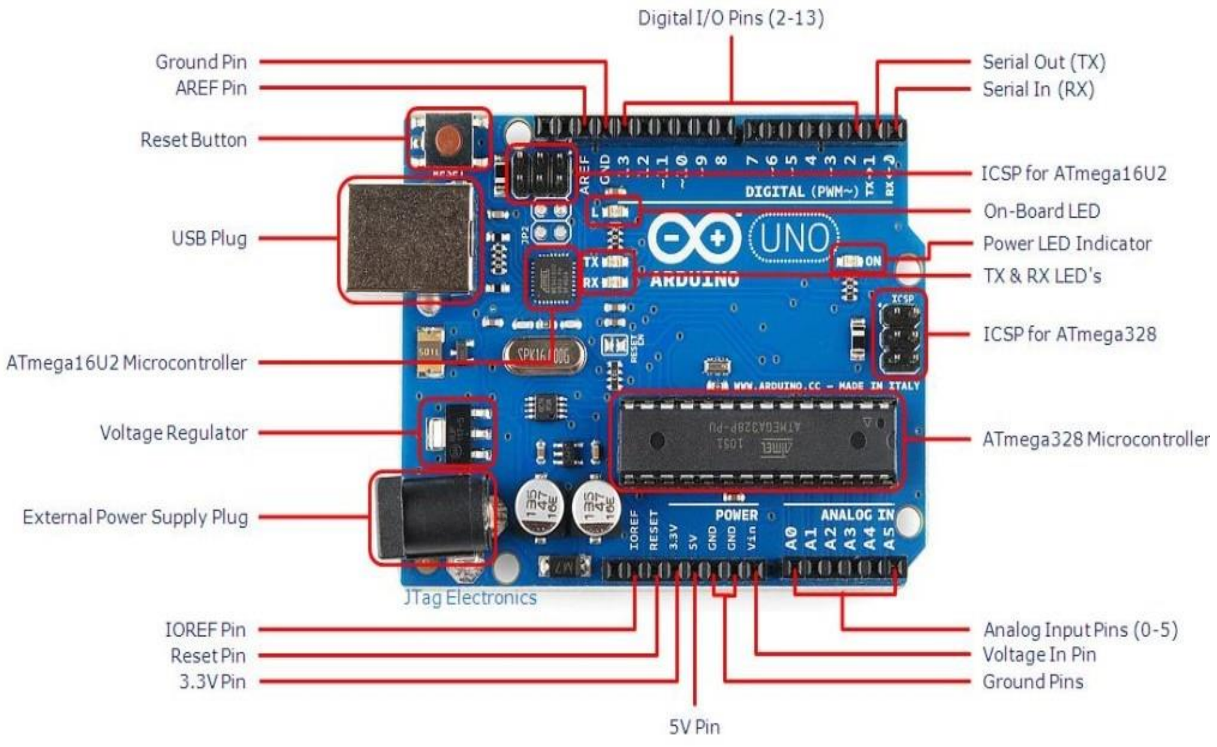

Figure 2-1 Arduino Uno R3 Pinout Diagram

Pin	Function	Type	Description
1	IOREF	IOREF	Reference for digital logic V - connected to 5V
2	Reset	Reset	Reset
3	+3V3	Power	+3V3 Power Rail
4	+5V	Power	+5V Power Rail
5	GND	Power	Ground
6	GND	Power	Ground
7	VIN	Power	Voltage Input
8	A0	Analog/GPIO	Analog input 0 /GPIO
9	A1	Analog/GPIO	Analog input 1 /GPIO
10	A2	Analog/GPIO	Analog input 2 /GPIO
11	A3	Analog/GPIO	Analog input 3 /GPIO
12	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
13	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line

Table 2-2 Analog pins of Arduino Uno R3

Pin	Function	Type	Description
1	D0	Digital/GPIO	Digital pin 0/GPIO
2	D1	Digital/GPIO	Digital pin 1/GPIO
3	D2	Digital/GPIO	Digital pin 2/GPIO
4	D3	Digital/GPIO	Digital pin 3/GPIO
5	D4	Digital/GPIO	Digital pin 4/GPIO
6	D5	Digital/GPIO	Digital pin 5/GPIO
7	D6	Digital/GPIO	Digital pin 6/GPIO
8	D7	Digital/GPIO	Digital pin 7/GPIO
9	D8	Digital/GPIO	Digital pin 8/GPIO
10	D9	Digital/GPIO	Digital pin 9/GPIO
11	SS	Digital	SPI Chip Select
12	MOSI	Digital	SPI1 Main Out Secondary In
13	MISO	Digital	SPI Main In Secondary Out
14	SCK	Digital	SPI serial clock output
15	GND	Power	Ground
16	AREF	Digital	Analog reference voltage
17	A4/SD4	Digital	Analog input 4/I2C Data line (duplicated)
18	A5/SD5	Digital	Analog input 5/I2C Clock line (duplicated)

Table 2-3 Digital pins of Arduino Uno R3

2.1.4 Board Outline & Mounting Holes

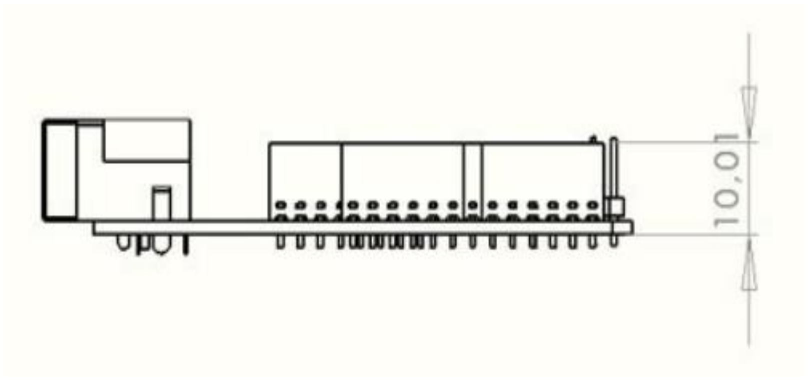

Figure 2-2 Side of the board

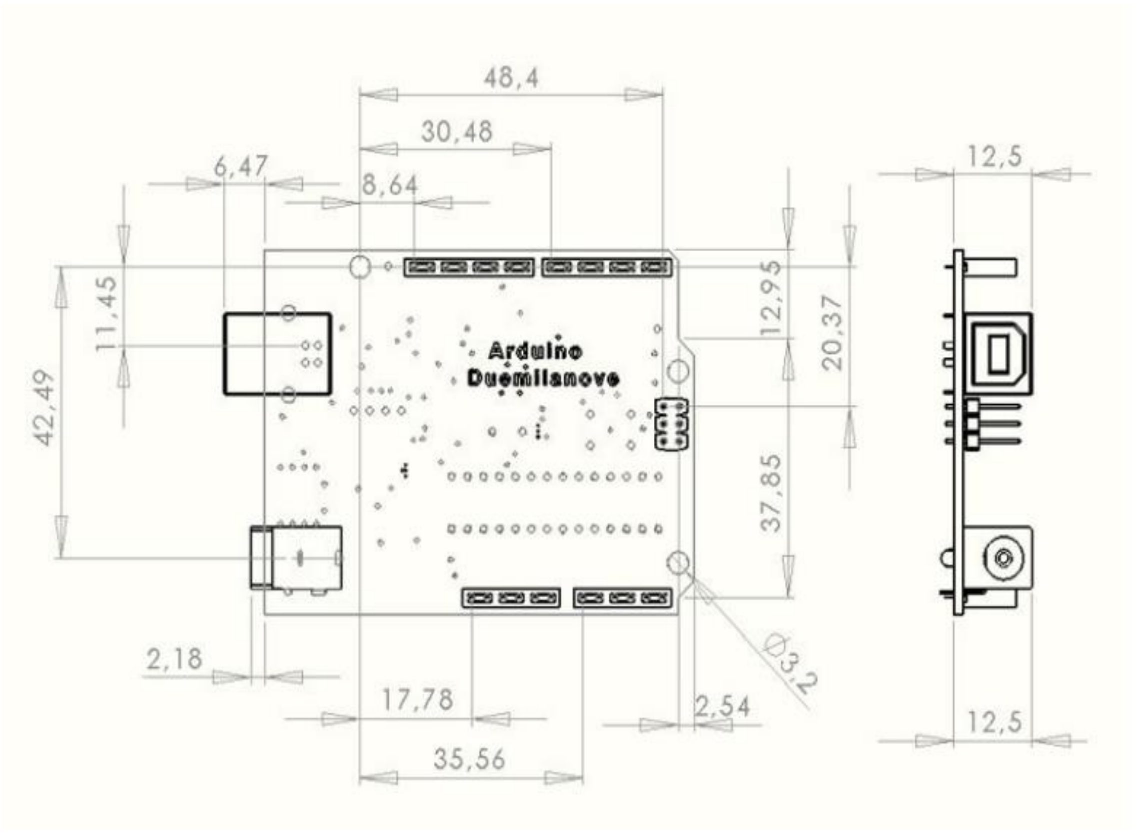

Figure 2-3 Front side of the board

2.2 L298N motor driver modules

The L298N module is a popular dual H-bridge motor driver that is ideal for driving DC motors with a microcontroller such as the Arduino. It employs the STMicroelectronics L298N motor driver IC, which can handle up to 2A continuous current per channel and 3A peak currents.

This module's primary features include:

- 2 DC motors or 1 stepper motor are controlled.
- The operating voltage range is 5V to 35V.
- 5V logical voltage, suitable for microcontrollers
- Flyback diodes are built-in to protect against motor back EMF.
- Motors can be turned on and off using enable/disable pins.
- PWM can be used to regulate the speed of motors via output pins.

To drive the motors, the L298N module requires a separate power supply, while the logic is still powered by the Arduino at 5V. It makes it quite simple to begin controlling motors in your electronics projects.

The L298N module, with its small size, screw connections, and simple control pins, is an excellent choice for adding motor capabilities to your DIY robotics or automation projects. It's inexpensive and simple to use with Arduino or other microcontroller boards.

Overall, the L298N motor driver module is a useful and versatile module for driving small DC motors in a variety of hobbyist and educational applications.

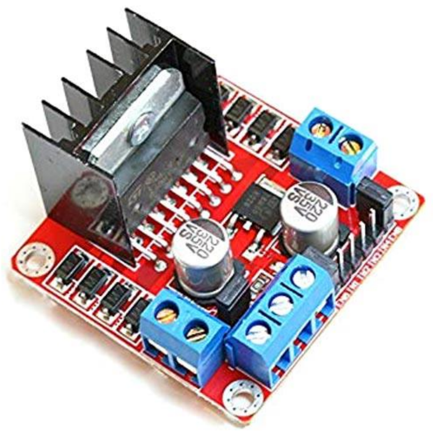

Figure 2-4 L298N Motor Driver Module

2.2.1 Controlling a DC Motor

We can only have complete control over a DC motor if we can control its speed and direction of rotation. Combining these two strategies makes this achievable.

- PWM is used to control speed.
- The H-Bridge is utilized to control the rotation direction.

2.2.1.1 PWM is used to control speed.

A DC motor's speed can be changed by varying its input voltage. Pulse Width Modulation (PWM) is a popular approach for accomplishing this.

PWM is a technique for adjusting the average value of an input voltage by transmitting a series of ON-OFF pulses. The average voltage is proportional to the pulse width, which is known as the Duty Cycle.

The higher the duty cycle, the higher the average voltage applied to the DC motor, causing the motor speed to increase. The lower the average voltage applied to the DC motor, the lower the duty cycle, and hence the motor speed.

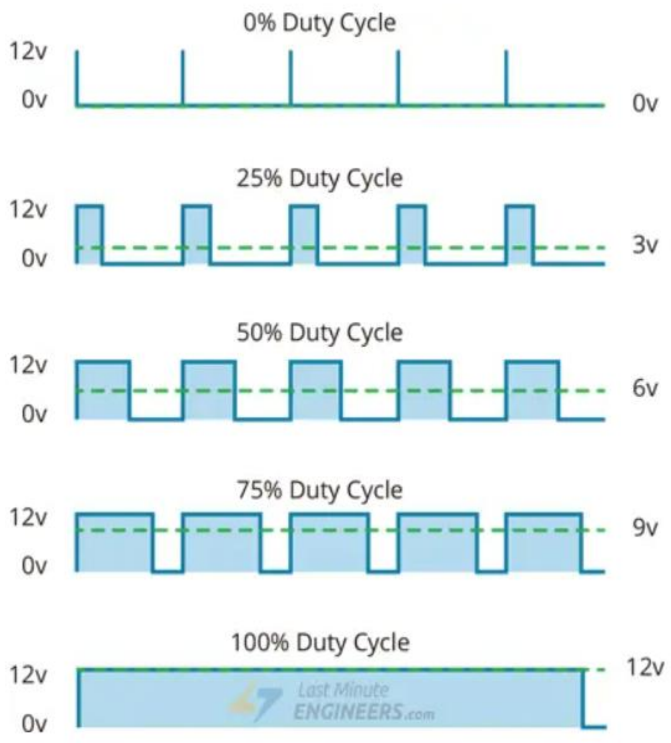

Figure 2-5 Pulse Width Modulation (PWM) Technique

2.2.1.2 The H-Bridge is utilized to control the rotation direction.

The polarity of a DC motor's input voltage can be changed to influence its spinning direction. An H-bridge is a popular method for accomplishing this.

An H-bridge circuit consists of four switches organized in the shape of an H, with the motor in its center.

When two specified switches are closed at the same time, the polarity of the voltage provided to the motor is reversed. This causes the motor's spinning direction to change.

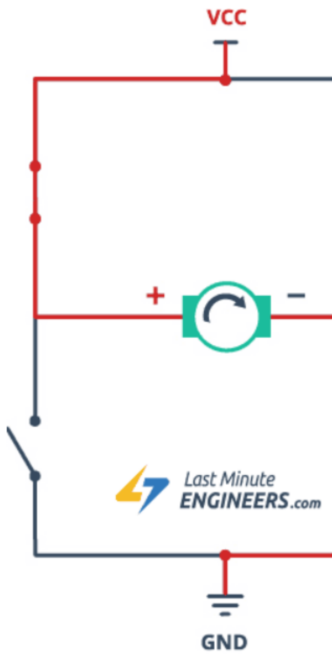

Figure 2-6 Working of H-Bridge

2.2.2 L298N Motor Driver Chip

The L298N is a large, black chip with a large heat sink in the center of the module.

The L298N chip has two standard H-bridges that can drive two DC motors, making it perfect for developing a two-wheeled robotic platform.

The L298N motor driver has a 5V to 35V supply range and a continuous current of 2A per channel, thus it works well with the majority of our DC motors.

2.2.3 L298N Module Pinout Configuration

The L298N motor driver module includes various input and output pins for controlling the linked motors. The IN1 and IN2 pins are Motor A's input pins, and they are used to control the spinning direction of Motor A. Similarly, IN3 and IN4 are Motor B's input pins that control its spinning direction. The ENA and ENB pins enable the PWM signal for Motors A and B, allowing for speed control. The OUT1 and OUT2 pins are connected to Motor A, while the OUT3 and OUT4 pins are connected to Motor B. To run the motors, the module requires a 12V input from a DC power supply. The internal switching logic is powered by the 5V pin.

2.2.4 Features & Specifications of L298N motor driver modules

The L298N dual H-bridge motor driver chip allows this module to control two DC motors or one stepper motor individually. It has a maximum motor supply voltage of 46V and can supply up to 2A continuous current per channel. The logic voltage is 5V, which makes it compatible with microcontrollers. The driver itself is powered by 5-35V.

The module has current monitoring for each motor and a heatsink to avoid overheating for optimal performance. When the status indicator is turned on, an LED illuminates.

This compact driver module, with a maximum power rating of 25W, is intended for applications working on up to 35V and requiring up to 2A per motor. This is a suitable solution for motor control in robotics, automation systems, and other mechatronics projects due to the combination of robust driver capacity, logical level changing, and integrated protection.

❖ **Brief about L298N Module**

The L298N Motor Driver module is made up of an integrated circuit that includes an L298 Motor Driver IC, a 78M05 Voltage Regulator, resistors, capacitors, a Power LED, and a 5V jumper.

Only when the jumper is inserted will the 78M05 voltage regulator be enabled. When the power source is less than or equal to 12V, the voltage regulator powers the internal circuitry, and the 5V pin can be utilized as an output pin to power the microcontroller. When the power source is more than 12V, the jumper should not be used, and a separate 5V should be supplied through the 5V connector to power the internal circuitry.

The ENA and ENB pins regulate the speed of Motor A and Motor B, respectively, whereas the IN1& IN2 and IN3& IN4 pins control the direction of Motor A and Motor B.

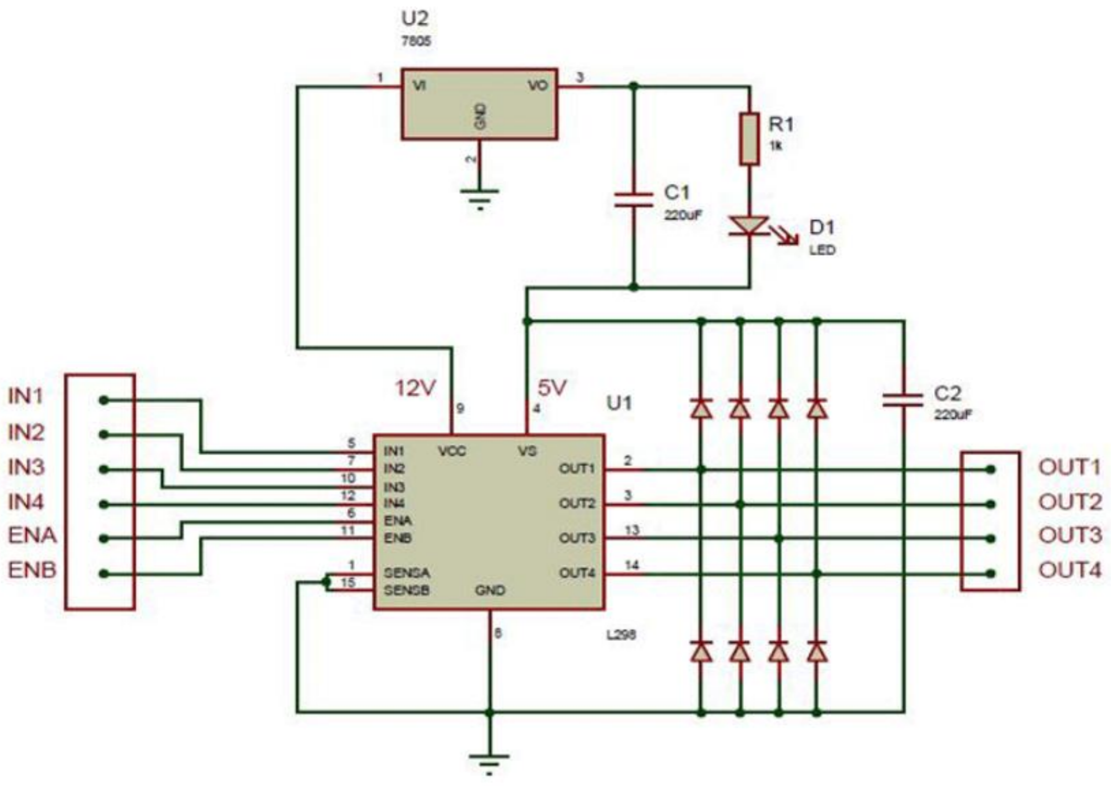

Figure 2-7 Internal circuit diagram of L298N Motor Driver module

2.3 Loadcell 1kg

2.3.1 What is Loadcell ?

A load cell is a sensor capable of converting force or weight into an electrical signal. The output value is proportional to the resistive strain change in the resistor bridge, resulting in a proportional voltage signal. The resistive load cell operates on the pressure-resistance principle. When a load or force acts on the sensor, the resistance changes. When an input voltage is applied, this change in resistance causes a corresponding change in output voltage.

Load cell sensors are used in a variety of applications in everyday life, including monitoring object weight, equally distributing product weight in automated production lines, weighing trucks, and so on.

The load cell is multifunctional, allowing for automated weight or force monitoring and control in a variety of industrial and commercial contexts. Because of its electrical interface, force data may be simply digitized and integrated into monitoring and control systems.

2.3.2 Structure of Loadcell

The load cell is made up of two parts: the strain gauge and the load. A load cell is typically made up of strain gauges bonded to the load cell body's surface. The load cell body is an elastic metal block that can be designed in a variety of shapes and materials (aluminum alloy, stainless steel, etc.) depending on the load cell type and application.

- The strain gauge is a characteristic resistor that changes resistance when compressed or stretched and is powered by a constant source.
- An elastic metal bar serves as the load.

In summary, strain gauges monitor the deformation of the load cell body caused by an applied force. This enables the load cell to transform mechanical force into a proportional electrical signal. The load cell construction converts applied force into quantifiable resistance change.

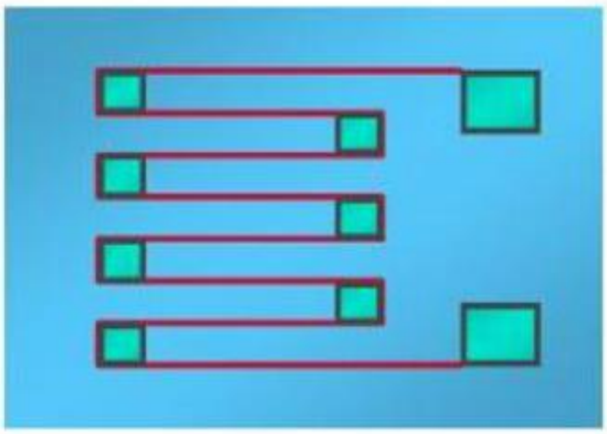

Figure 2-8 resistor circuit of the load cell

❖ The resistance R of a strain gauge is determined by:

$$R = \frac{\rho * l}{S}$$

Where:

R = Resistance of the strain gauge (Ohms)

L = Length of the strain gauge metal wire (m)

S = Cross sectional area of the metal wire (m2)

When a force is applied to the wire, the resistance changes.

- When a force is applied to a wire, the length L lowers and the resistance falls.
- When a force is applied to a wire, the length L rises and the resistance increases.

As a result, the resistance changes proportionally to the applied force. This connection is used by the strain gauge to transform an applied force into a measured change in electrical resistance. This is the basic operating concept that enables load cells to detect weights and forces.

2.3.3 Operating principle

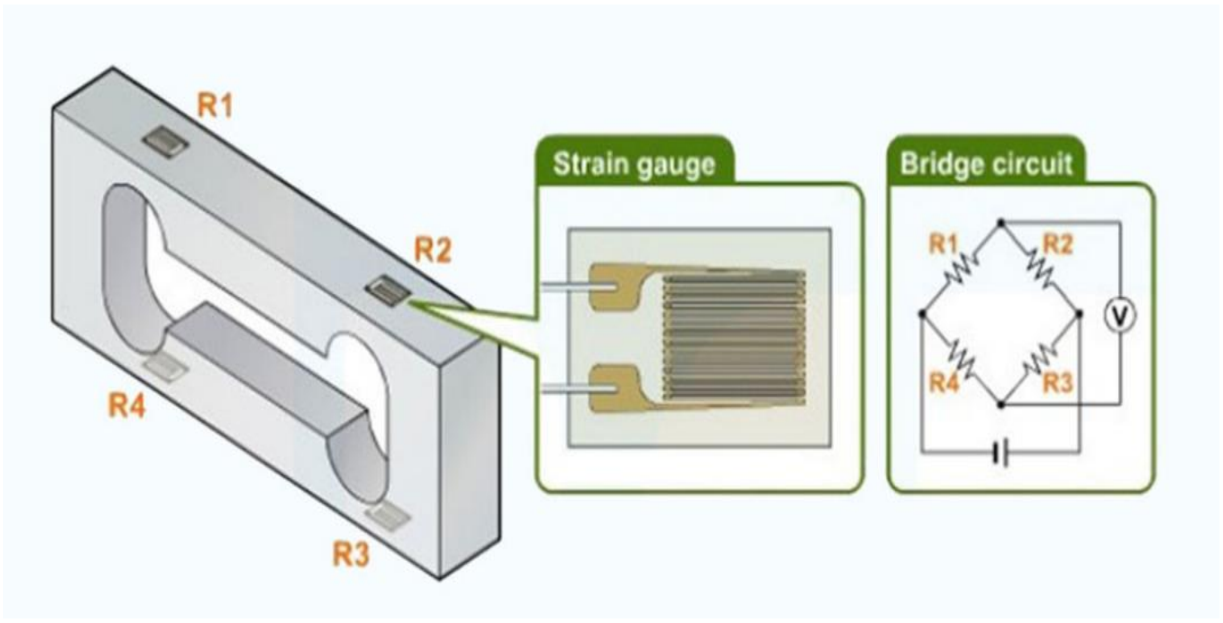

Figure 2-9 Figure Structure of Loadcell

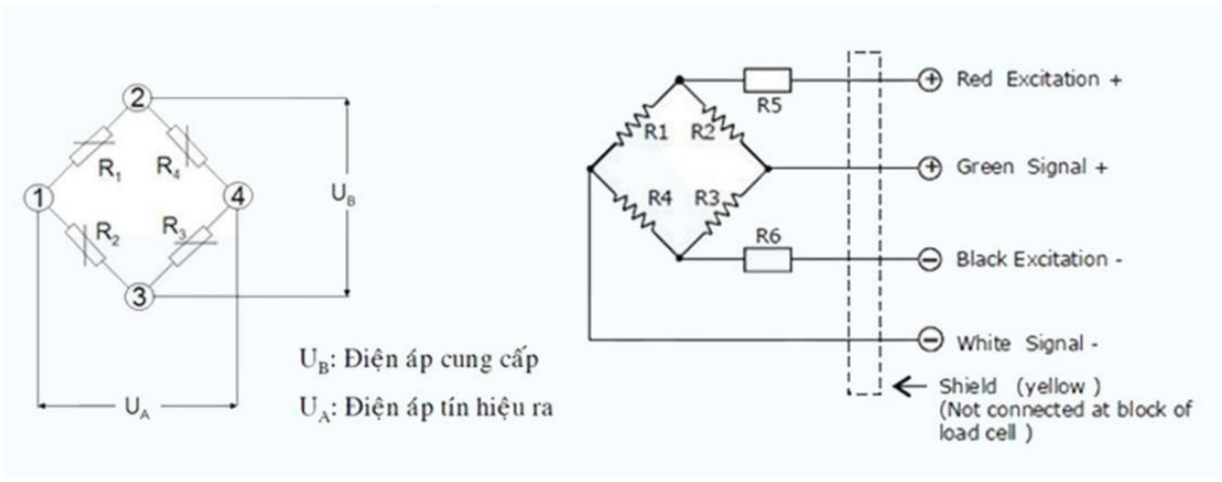

Figure 2-10 Figure Structure of Wheatstone bridge circuit

A voltage is applied to the load cell's input (at Wheatstone bridge circuit corners (1) and (4)), and the output signal voltage is measured between the other two corners.

When the four resistors are matched in value, the output signal voltage is zero or close to zero in the balanced condition (unloaded state). When a load or force is applied to the load cell body, causing deformation (tension or compression), the length and cross-sectional area of the metal wires of the strain gauge resistors change, which changes the resistance values and hence the output voltage.

Note: Sudden activities, such as abruptly placing or hurling a weighed object onto the weighing platform, might induce sudden deformation of the metal bar, resulting in erroneous readings and possibly Strain Gauge damage. Avoid weighing objects with weights that are much more than the load cell's measurement range.

In summary, the load cell uses strain gauge resistance variations in the Wheatstone bridge circuit to transform an applied weight/force into a measured electrical output. Loading should be done with caution to ensure accuracy and prevent sensor damage.

2.3.4 Specifications of loadcell

The YZC-133 electronic scale load cell has a rated output of 1.0 ± 0.15mV/V and weighs 1Kg. It has an input resistance of 1066 ± 20Ω and a nonlinearity of 0.05%. The resistance at the output is 1000 ± 20Ω. This load cell operates at 5 volts and has an operational temperature range of -20 to 65 degrees Celsius. The sensor is constructed of an aluminum alloy. It has a 180mm cable with wires in red, black, green, and white. The input connections are the red and black wires, with red being positive and black being negative. The output connections are the green and white wires, with green being positive and white being negative. In conclusion, the key characteristics of this 1Kg load cell make it well-suited for electronic scale and other weighing applications needing precision force measurement up to 1Kg loads.

2.4 Encoder 400 pulses 5-24VDC

A device that translates motion or location into electrical signals for measurement and control is known as an encoder. Rotary and linear encoders are the most frequent types.

Rotational motion is measured using rotary encoders. The encoder creates digital or analog output signals when the shaft rotates, showing the rotation direction, number of shaft revolutions, or angular position. Some examples of common rotary encoders are:

- Optical encoders detect rotation by using an LED, a photodetector, and a disk containing patterns.

- Magnetic encoders detect changes in magnetic fields to determine rotation.

- Contacts on mechanical encoders open and close as the shaft rotates.

Linear encoders are devices that measure linear motion. To calculate position, speed, and direction, a read head detects movement of a scale or strip. Linear encoders are used in metrology, CNC machines, manufacturing, and other high-precision linear positioning applications.

Closed-loop feedback is provided by encoders in motor control systems. Encoder output signals are utilized for accurate speed regulation, position control, synchronization, and other purposes. Many automated motion and control systems rely on encoders.

Encoders convert motions into useable electrical impulses. This enables precise monitoring and control of speed, position, and other parameters in automation equipment and industrial processes in real time.

2.4.1 Overview of the basic structure and components of a rotary encoder:

- Shaft - The encoder's rotating component. The encoder senses rotation as the shaft rotates.

- Code wheel/disc - This disc is fastened to and spins with the shaft. It is carved with designs such as slots, holes, or lines.

- Light source - A light source, such as an LED or infrared LED, that shines light onto the code wheel.

- Photodetector - A photodiode or phototransistor that detects light that has been reflected or disrupted by code wheel patterns.

- Housing - This component houses the internal components and is connected to the shaft. The encoder is shielded from dust, filth, and damage.

- Bearings allow the shaft and code wheel within the housing to rotate smoothly and precisely.

- Electrical interface - The connections and electronics that transfer the photodetectors' output signals.

- Output signals - An incremental encoder's output signals are digital pulses that carry information about the shaft's rotation direction and the number of increments turned.

In summary, the fundamental components allow the encoder to capture rotational motion properly and transform it into digital or analog electrical data. The enclosure protects the motor from mechanical damage, while the bearings allow for smooth, low-friction running.

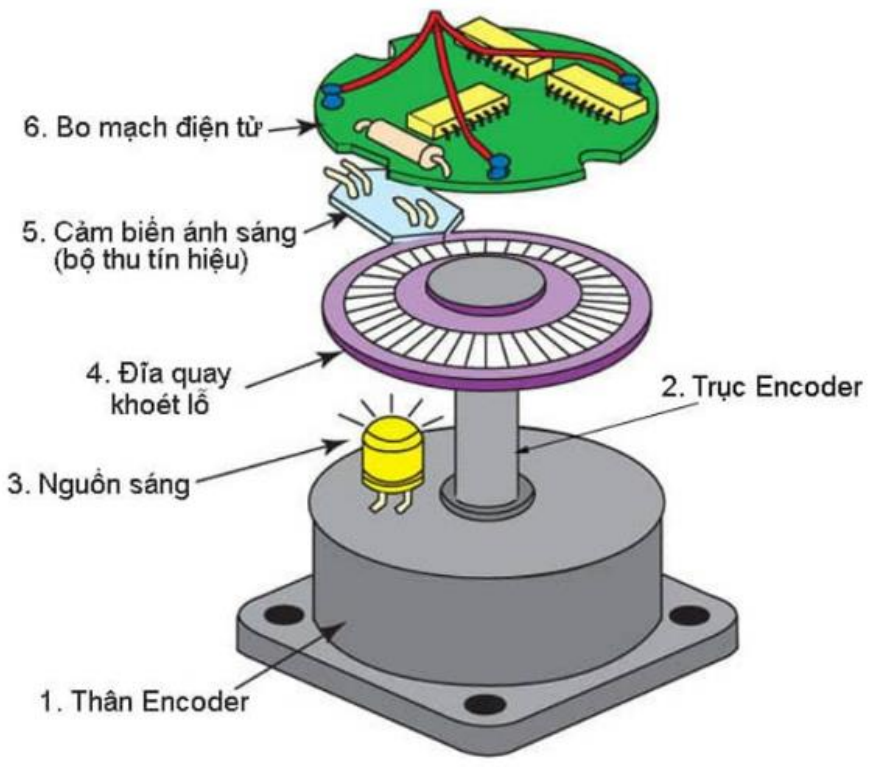

Figure 2-11 Basic structure and components of a rotary encoder

2.4.2 What are the basic types of Encoders?

Encoders are used to track the angular position of a rotating disc, such as a wheel, motor shaft, or any device requiring angular position determination.

Absolute encoders and incremental encoders are the two primary types of encoders based on their output.

An absolute encoder generates a unique digital code for each degree of shaft rotation. This ensures that the precise angular location is always known.

An incremental encoder generates pulse signals that indicate motion increments rather than exact location values. To measure relative displacement from a reference location, the pulses can be counted.

	Absolute encoders	**Incremental encoders**
Advantages	- Higher overall resolution than the relative type. - Better start-up performance. - Accurate multi-axis motion detection. - Many output protocols. - Better recovery after system failures, able to remember lost power positions because each position has its own signal code	- Easy to process return signals. - Flexible design suitable for many applications. - Fewer related sensors so the system is simpler and cheaper compared to absolute types
Disadvantages	- High cost due to complex manufacturing - Harder to read signals compared to the relative type	- Easy to pulse error when returning signals - Long-term operation accumulates more errors over time

Table 2-4 Comparison between absolute encoders and incremental encoders

Absolute encoders offer absolute position data, whereas incremental encoders provide relative motion and direction data. The option is determined by the location tracking and control requirements of the application.

2.4.3 Operating principle of an encoder:

An encoder's operation is based on turning motion into an electrical signal. A light source, a photodetector, and a coded disc or strip attached to a moving shaft or slide are used to do this.

The coded disc or strip interrupts or reflects light from the light source as the shaft rotates or the slide advances linearly. The photodetector detects this light and generates electrical pulses.

In an optical rotary encoder, for example, the coded disc has uniformly spaced lines engraved into its surface. Light pulses are formed as the lines disrupt the light beam as it rotates between the LED and photodiode. These pulses represent incremental rotation.

The relative angular location can be established by counting the pulses from a reference point. The rotational speed is indicated by the pulse frequency. Additional tracking markings are included in some encoders to establish absolute position.

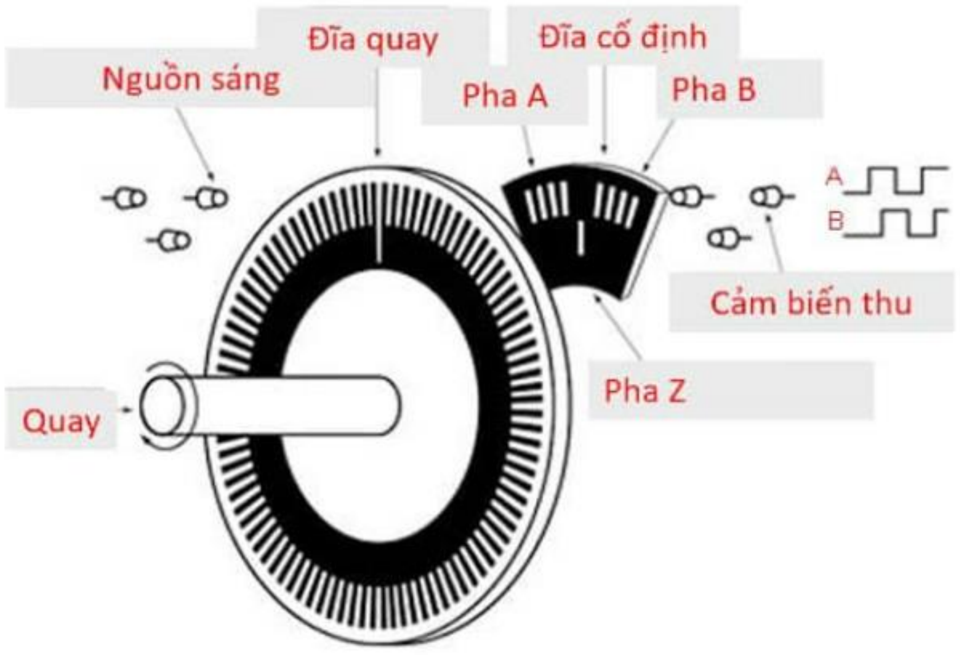

Figure 2-12 Operating principle of an encoder

In conclusion, the basic operational concept is:

1. Movement is the cause of disc/strip movement.

2. The pattern on the disc/strip disrupts or reflects light.

3. A photodetector detects light and creates electrical impulses.

4. Signals are analyzed to identify motion characteristics.

This clever optoelectronic technique enables encoders to give precise motion feedback for current control systems.

2.5 HX711 (24bit Loadcell ADC Converter Circuit HX711)

2.5.1 HX711 24-bit Load Cell ADC Converter Circuit Introduction:

In an introduction, I would emphasize the following crucial points:

- The HX711 load cell ADC converter circuit's purpose/function
- It transforms an analog voltage from a load cell into a 24-bit digital value.
- Key features include a 24-bit resolution and an 80 SPS conversion rate.
- To link the load cell(s), a wheatstone bridge design is used.
- Advantages over lower resolution ADCs, such as increased precision for weighing scales/systems
- The HX711 ADC converter circuit features excellent integration, fast response, noise immunity, and high reliability, in addition to a few fundamental functionalities.
- This is a circuit that takes the values from a load cell sensor with a 24-bit resolution and converts them to a 2-wire interface (clock and data) for data transmission to a microcontroller/Arduino.
- A summary of major components such as the HX711 IC, reference voltage, amplifier, and so on.

2.5.2 Function of pins of HX711.

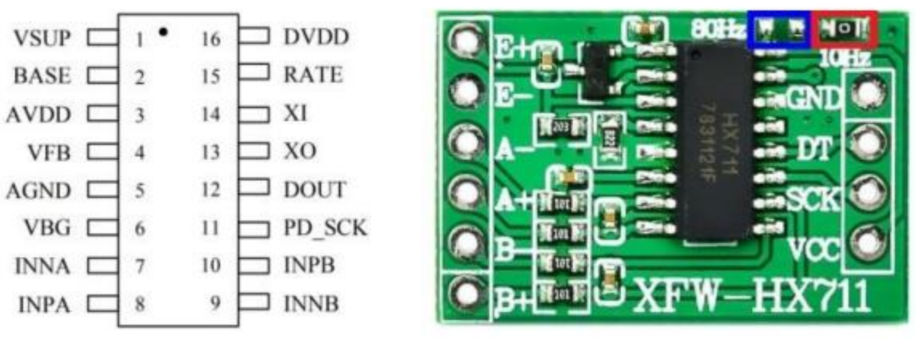

Figure 2-13 HX711 chip pinout and module HX711

No.	Name	Function	Operation
1	VSUP	Power Supply	Operating Range: 2.7 ~ 5.5V
2	BASE	Analog Output	Output Adjustment
3	AVDD	Power Supply	Analog Supply: 2.6 ~ 5.5V

4	VFB	Analog Input	Input Adjustment
5	AGND	Ground	Analog Ground
6	VBG	Analog Output	Bypass Reference Output
7	INA-	Analog Input	Channel A Negative Input
8	INA+	Analog Input	Channel A Positive Input
9	INB-	Analog Input	Channel B Negative Input
10	INB+	Analog Input	Channel B Positive Input
11	PD_SCK	Digital Input	Power Down and Serial Clock Input Control
12	DOUT	Digital Output	Serial Data Output
13	XO	Digital I/O	Crystal I/O
14	XI	Digital Input	Crystal I/O or External Clock Input, 0: Use On-Chip Oscillator
15	RATE	Digital Input	Output Data Rate Control, 0: 10Hz; 1: 80Hz
16	DVDD	Power Supply	Operating Range: 2.6 ~ 5.5V

Table 2-5 Describe the functional diagram of the pins of the HX711

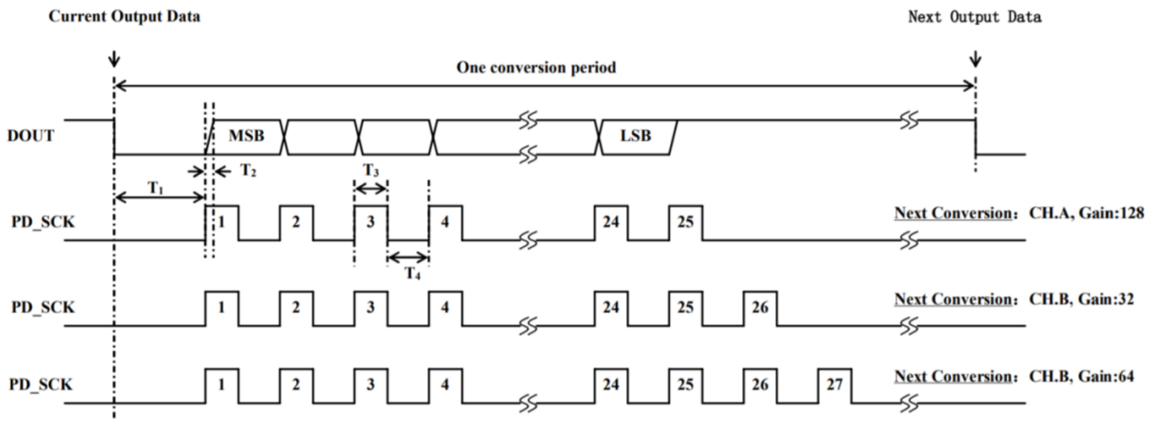

Figure 2-14 Data reading and gain selection timing

Symbol	Operation	MIN	TYP	MAX	Units
T1	DOUT fall to PD_SCK rise	0.1			µs
T2	PD_SCK rise to DOUT data ready			0.1	µs
T3	PD_SCK high time	0.2	1	50	µs
T4	PD_SCK low time	0.2	1		µs

Table 2-6 Operational process of input data and output data.

2.5.3 Features & Specifications of HX711

- Operating voltage: 2.7 – 5V
- Current consumption: < 1.5 mA
- Sampling rate: 10 – 80 SPS (adjustable)
- Resolution: 24 bit ADC
- Voltage resolution: 40mV
- Size: 38 x 21 x 10 mm

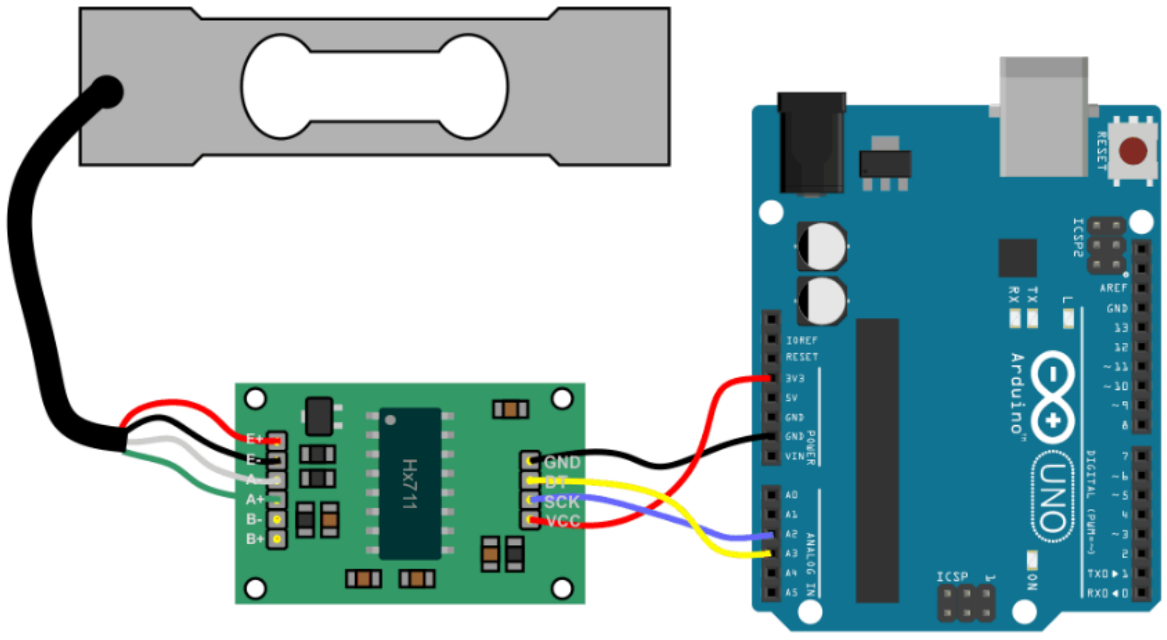

Figure 2-15 Connection diagram of 24-bit adc converter circuit with arduino uno

CHAPTER 3. Design block diagrams and principle diagrams

3.1 Block diagram of the system

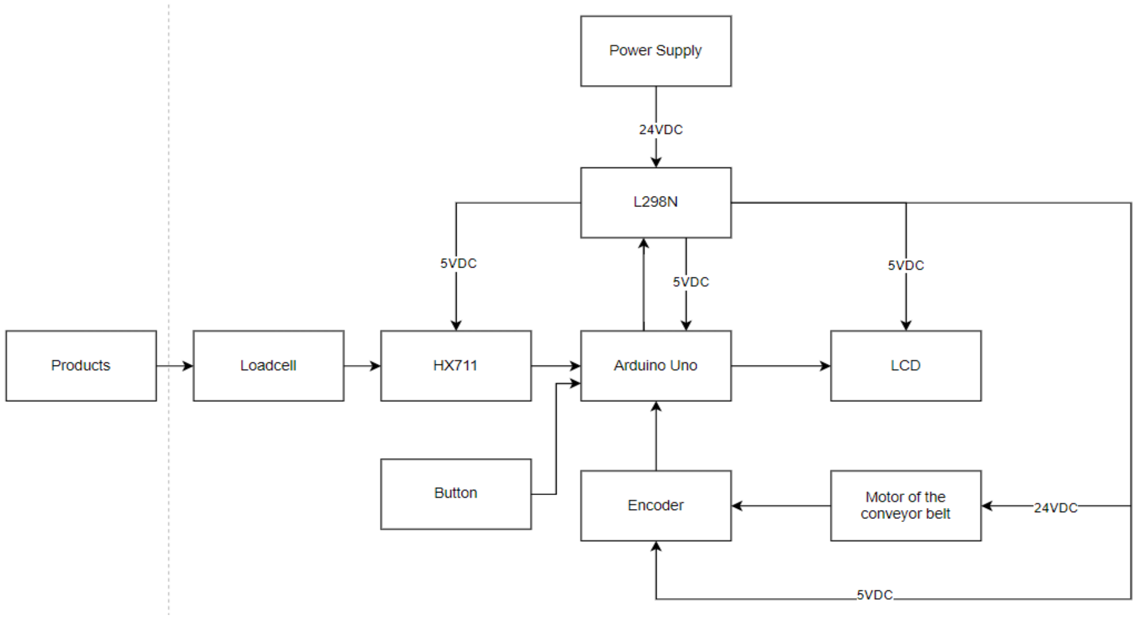

Figure 3-1 Block diagram of the system

3.2 Principle diagrams of the system

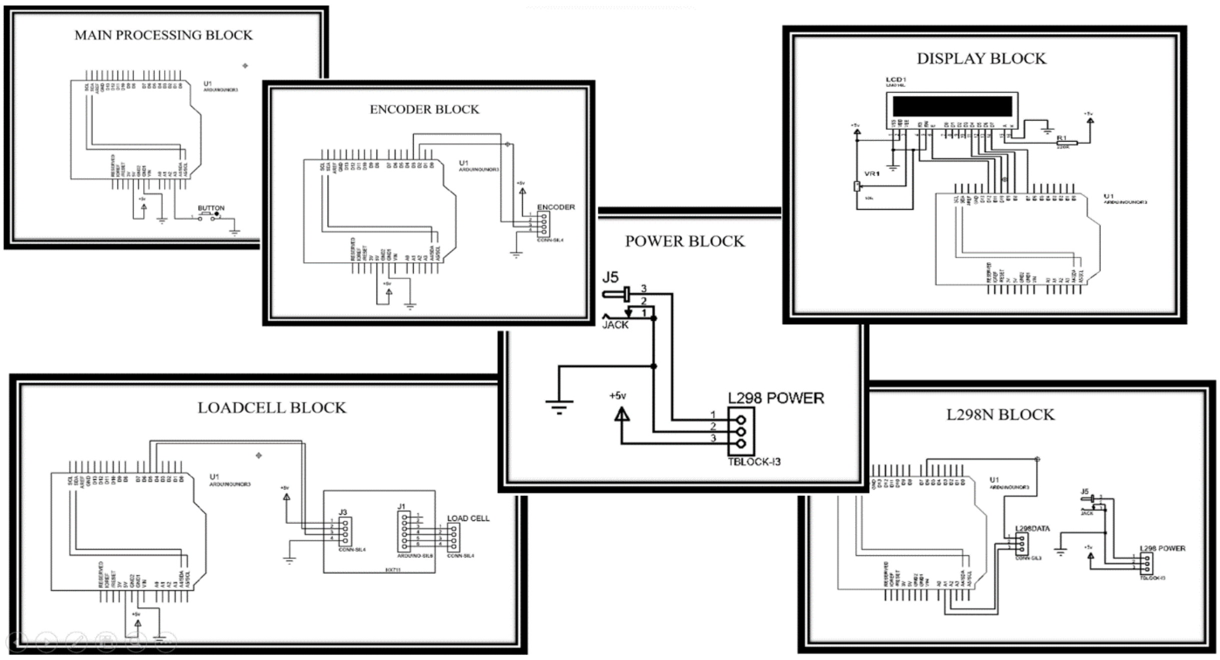

Figure 3-2 Principle diagrams of the system

3.2.1 Power supply block

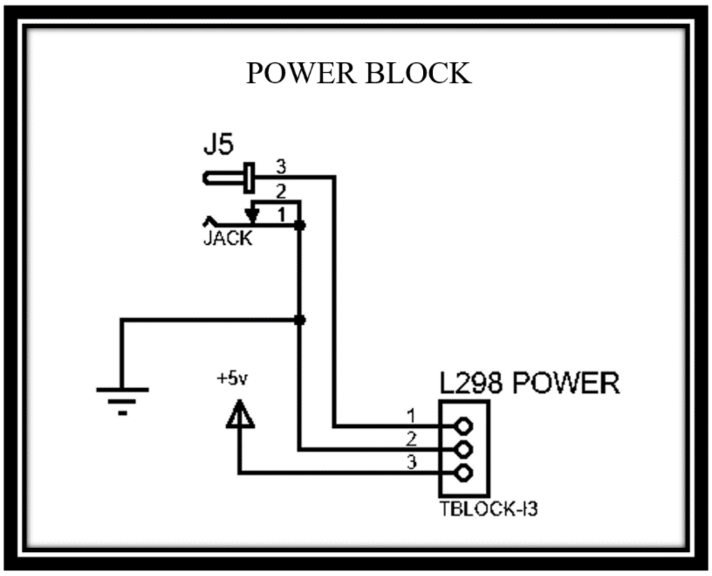

Figure 3-3 Power supply block

Here's how to get the 5VDC electricity from the L298N motor driver module to power other modules:

1. Check the L298N datasheet carefully to determine the precise pin placement of the 5VDC output. It is commonly labeled 5V or Vcc.

2. Use a power adapter or battery with a voltage output range of 7-35VDC. The more voltage there is, the more 5VDC current it can supply. Connect the power source's positive and negative terminals to the L298N module's two power input pins (VCC and GND).

3. Connect the L298N's 5VDC output pin to the power input pin (VCC or 5V/3.3V) of the module that requires power using jumper cables.

4. Connect the common Grounds (0V) between the L298N and the powered module. This completes the electrical circuit.

5. Check the 5V output voltage with a multimeter. The value should be steady at 4.95-5.05V.

6. Adjust the L298N power supply to keep the output voltage steady. Also, do not exceed the maximum current that the 5VDC pin can supply.

This is how you can get a steady 5V supply for various modules from the L298N motor driver module, without needing additional power sources. You can use this to power Arduino modules, sensors, or other electronics within the current budget.

Note that the supply current from the 5VDC source of the L298N module is usually quite small, around 500mA. Therefore, you should only use it to power small modules, not connect motors or other high-power electrical loads directly.

If you need to supply more than 500mA of 5V current, you need to add a step-down voltage regulator module such as the LM2596 buck converter module. This can convert higher voltages to stable and higher-current 5V to power more demanding components and circuits.

3.2.2 Display block

- **LCD1602 Screen**

A very common 16x2 alphanumeric liquid crystal display

Can display 32 characters at a time in 2 rows

Requires power supply of 4.7-5.3V

- **Arduino Uno**

Popular microcontroller development board

Has 14 digital I/O pins to connect to various devices

Provides 5V power from its 5V pin

- **10kΩ Variable Resistor**

Used to adjust LCD's contrast ratio

Connected in series between 5V pin and VO pin of LCD

Rotating knob changes resistance from near 0 to 10kΩ

Adjust till text on LCD is clearly visible

- **220Ω Resistor**

Current limiting resistor for the backlight LED

Connected in series between 5V pin and backlight (BL) pin

Protects LED from excess current damage

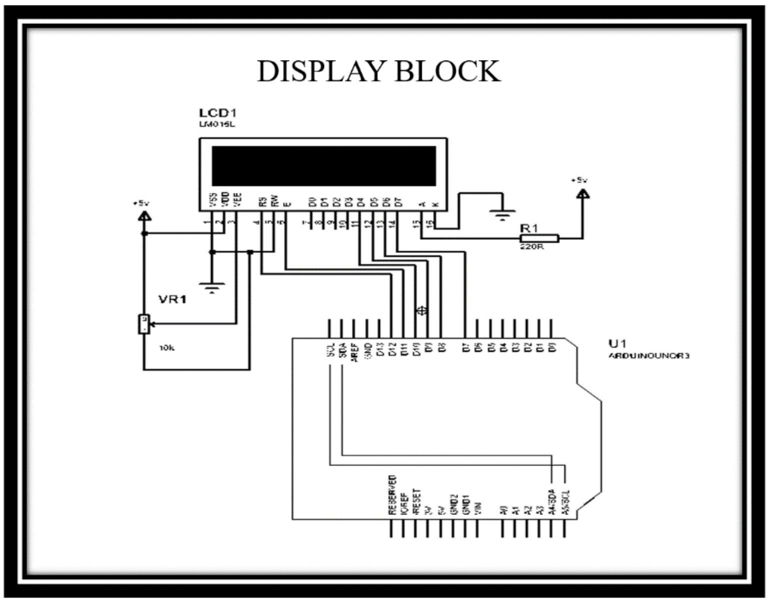

Figure 3-4 Display block

Together, these components enable text display on the LCD1602 which can be programmed via the Arduino board based on variable inputs and logic requirements.

3.2.3 L298N block

Here's an overview of how to connect the L298N motor driver module to an Arduino Uno:

❖ **Motor Driver Module L298N**

- Allows Arduino or microcontrollers to operate DC motors.

- Supports higher voltage and current than most controllers.

- Equipped with two H-bridge driver circuits for driving two DC motors individually.

- Requires a separate power supply of up to 46V to power the motors.

❖ **Arduino Connection**

- The L298N features two Enable/Input Pins that are used to control the two bridges.

- These are connected to the digital pins on the Arduino to enable motor motion.

- Additional pins govern the rotational direction of the motor based on the logic level.

- Power for logic is provided by connecting the Arduino's 5V pin to the L298N's 5V pin.

- The circuit is completed by connecting the GND pins together.

To operate a motor, Arduino delivers HIGH or LOW signals to input pins, instructing the L298N which direction to spin the motors and turning on the bridges. The motors themselves are powered by an external source.

This configuration allows two DC motors to be controlled separately in both directions using basic digital outputs from Arduino.

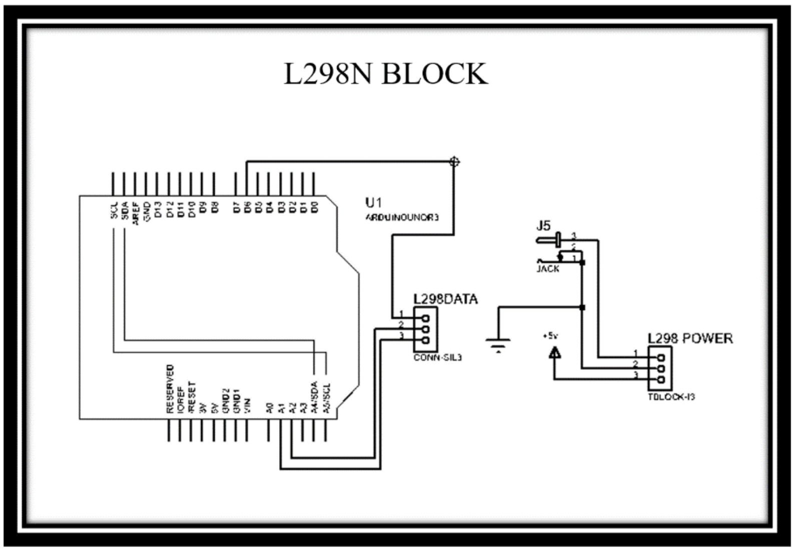

Figure 3-5 L298N block

3.2.4 Ecoder block

Here's how to hook up a 2-phase encoder with 400 pulses per rotation to an Arduino Uno:

❖ **Encoder**

- A mechanical device capable of converting rotating motion into digital pulses.

- This device is used to precisely measure rotation speed, position, and direction.

- This encoder generates 400 pulses every complete revolution.

- Has two square wave outputs (Phase A and Phase B) that are 90 degrees apart.

❖ **Arduino Connection**

- The Phase A and Phase B lines are connected to the digital input pins of the Arduino.

- The pins can detect and count RISING and FALLING pulses.

- The Arduino can determine rotation direction by counting and tracking A vs B.

- Encoder and Arduino are linked via power and ground pins.

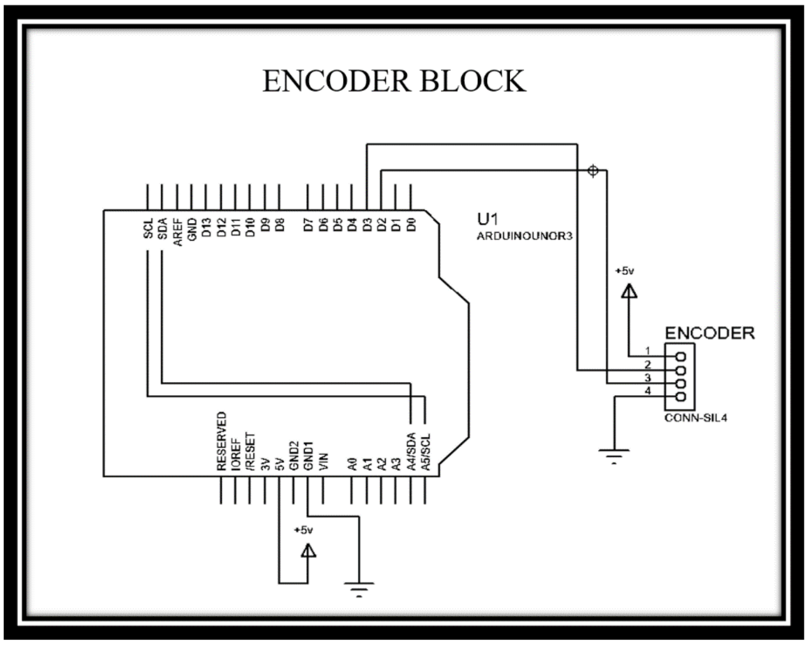

Figure 3-6 Encoder block

Arduino's code employs interrupts to count pulses efficiently and swiftly without interfering with the main program flow. The following is calculated by mathematical pulse counting algorithms:

- RPM of the motor

- Exact position determined by total pulse count since start

As a result of this link, the Encoder can send crucial rotary motion data to the Arduino for control systems.

3.2.5 Loeadcell block

Here's an overview of how to connect a 1kg load cell to an Arduino Uno using the HX711 module:

❖ **Loadcell**

- A transducer is a device that transfers a force like tension, compression, or pressure into an electrical signal.

- This is a 1kg model that provides accurate weight measurements up to 1kg.

- Extremely low voltage outputs necessitate the use of specialist circuitry.

❖ **The HX711 Module**

- High-precision 24-bit ADC module for weigh scales and load cells.

- The instrumentation amplifier provides precise voltage readings.

- A programmable gain amplifier, voltage reference, and ADC are built in.

- Arduino and 3.3/5V logic interface compatibility

❖ **Connections:**

- The load cell includes four strain gauge wires that link to the HX711

- The HX711 outputs connect to the digital pins on the Arduino

- 5V and Ground pins are wired together between modules.

❖ **To determine weight:**

1. The HX711 amplifies and converts the voltage of the load cell into 24-bit ADCs.

2. The ADC value is read by Arduino via the serial interface.

3. The formula computes the actual weight depending on the calibration.

This configuration enables an Arduino system to precisely measure weights placed on a load cell.

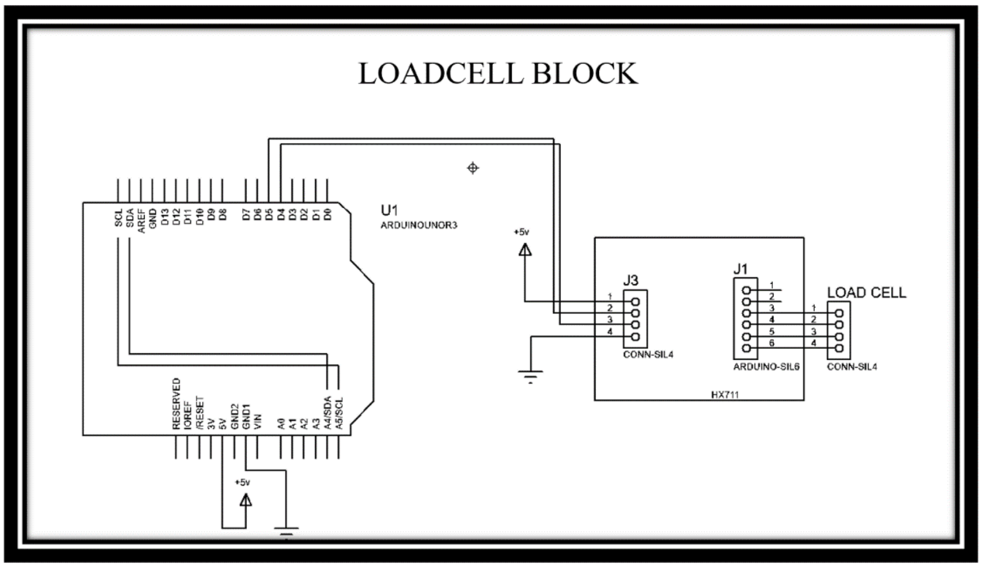

Figure 3-7 Loadcell block

3.3 Printed circuit board (PCB) circuit

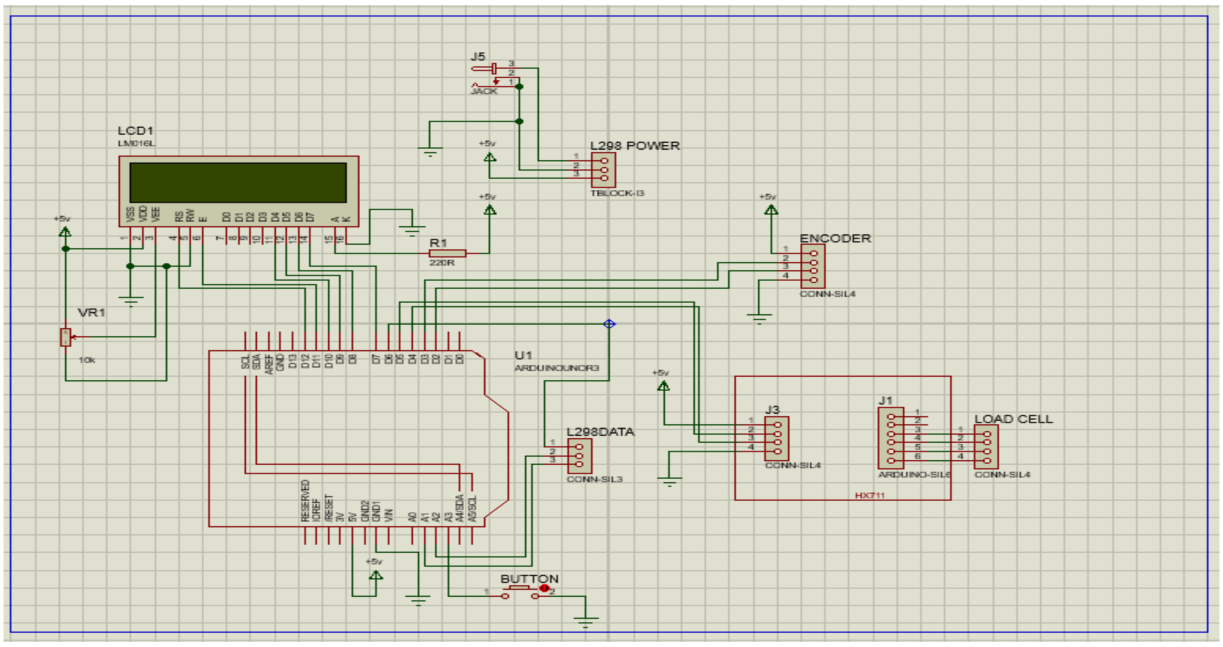

Figure 3-8 Simulate the principle diagram using Proteus software

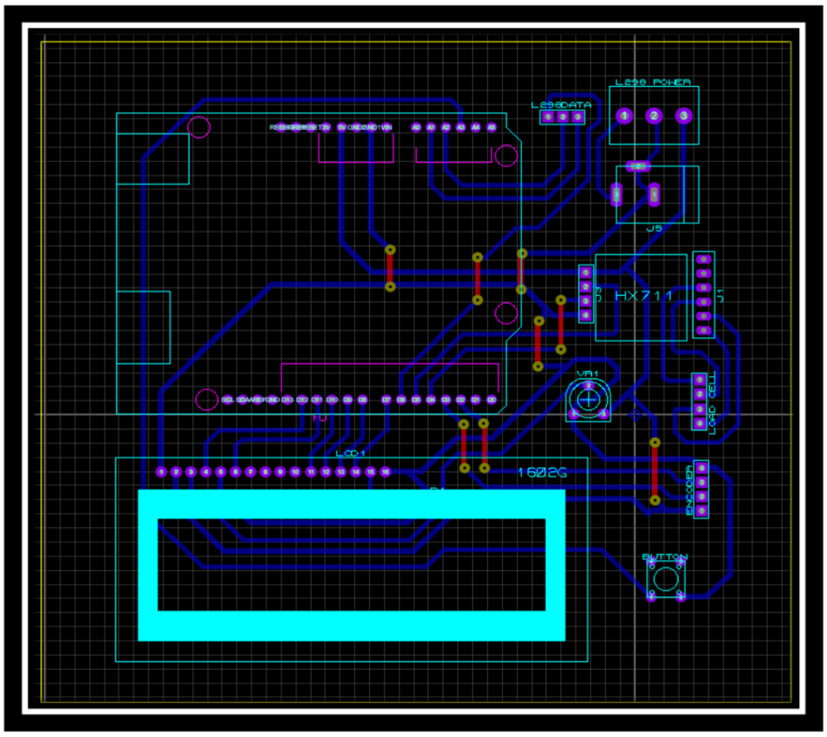

Figure 3-9 Printed circuit board

3.4 Flow chart

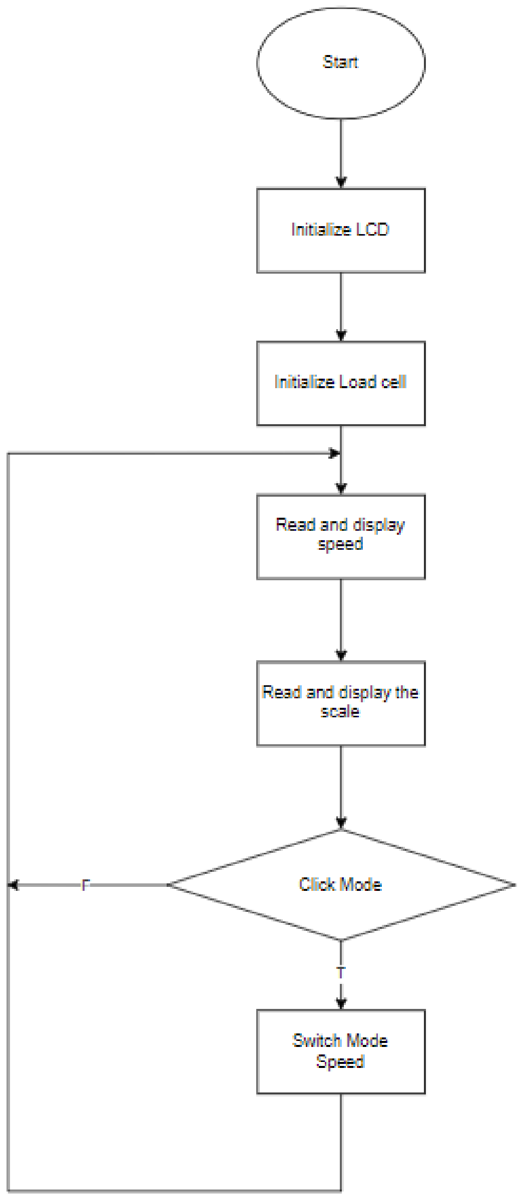

Figure 3-10 Flowchart of the system

3.5 System model

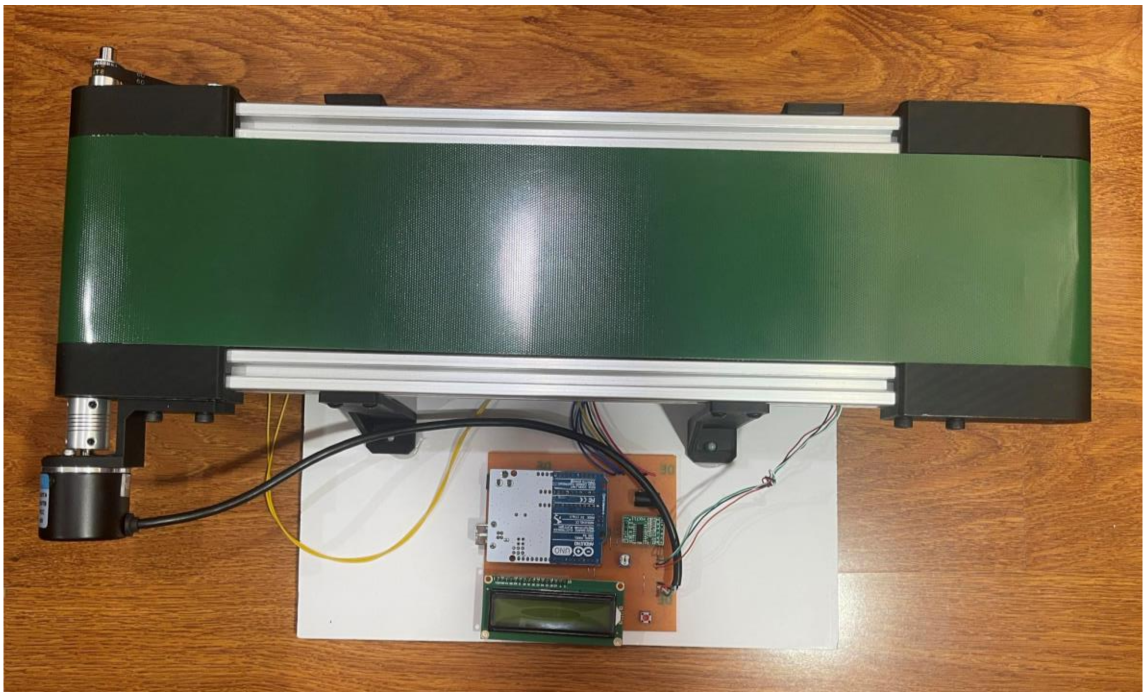

Figure 3-11 System module

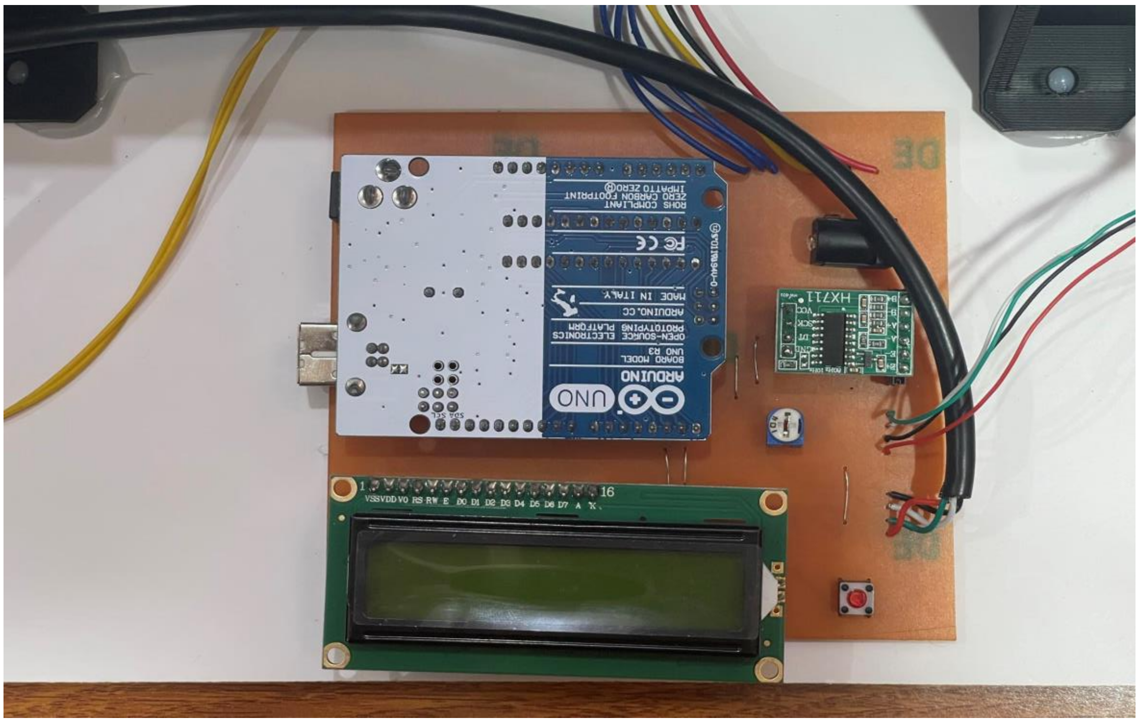

Figure 3-12 Printed cỉcuit

CHAPTER 4. CARRY OUT EXPERIMENT

4.1 Experimental process

Step 1: Supply power using a 24VDC-3A adapter

Step 2: Press the Mode button

Step 3: Place items weighing 50g on the conveyor belt

Step 4: Place items weighing 70g on the conveyor belt

Step 5: Take turns placing items of different weights on the conveyor belt

Step 6: Repeat steps 2,3,4 until mode 4

Step 7: Press the Mode button to Mode 0 to stop the conveyor belt

Step 8: Disconnect the power supply from the circuit

4.2 Experimental results

4.2.1 Load cell measurement capability test results:

- Measurement range: 0-1kg
- Accuracy: ±0.05kg
- Repeatability: 0.005kg

4.2.2 Speed measurement capability test results:

- Measurement range: 0-90 rpm
- Accuracy: ±5rpm

4.2.3 Speed control capability test results:

- Number of speed levels: 4 (45rpm, 60rpm, 75rpm, 90rpm)
- Response time: <1s

4.2.4 Stability assessment:

- Load fluctuation at steady state: <0.02kg
- Speed fluctuation at steady state: <3rpm

4.2.5 System performance evaluation:

- Power consumption: 24W (motor power not included)

In summary, the dynamic weighing system meets measurement, control, and stability requirements. The system runs continuously for lengthy periods of time with minimal energy use.

These are some examples of possible experimental results for the system. Depending on the planned use and real requirements, further acceptable test results may be included.

4.3 Experimental results survey table

4.3.1 Survey weighing 50g products with different speeds

❖ Mode 1:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	45	47	46	48	48	45	46	47	47	44	**46.3**
Speed (m/s)	0.061	0.064	0.063	0.065	0.065	0.061	0.063	0.064	0.064	0.060	**0.063**
Weight (g)	35	32	32	36	34	33	36	32	34	33	**33.7**

Table 4-1: 50g product weighing results with Mode 1.

❖ Mode 2:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	58	59	57	59	60	57	61	59	58	62	**59**
Speed (m/s)	0.079	0.080	0.078	0.080	0.082	0.078	0.083	0.080	0.079	0.084	**0.080**
Weight (g)	30	32	34	33	31	33	34	35	32	31	**32.5**

Table 4-2: 50g product weighing results with Mode 2.

❖ Mode 3:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	74	76	77	77	78	77	73	75	76	74	**75.7**
Speed (m/s)	0.101	0.103	0.105	0.105	0.106	0.105	0.099	0.102	0.103	0.101	**0.103**
Weight (g)	30	29	29	28	32	28	30	30	29	31	**29.6**

Table 4-3: 50g product weighing results with Mode 3.

❖ Mode 4:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	92	93	94	91	94	90	89	90	91	89	**91.3**
Speed (m/s)	0.125	0.127	0.128	0.124	0.128	0.123	0.121	0.123	0.124	0.121	**0.124**
Weight (g)	24	26	27	25	24	23	25	26	25	27	**25.2**

Table 4-4: 50g product weighing results with Mode 4.

4.3.2 Survey weighing 70g products with different speeds

- Mode 1:

Times	1	2	3	4	5	6	7	8	9	10	Averager
Speed (rpm)	48	47	45	47	46	46	45	46	47	45	**46.2**
Speed (m/s)	0.065	0.064	0.061	0.064	0.063	0.063	0.061	0.063	0.064	0.061	**0.063**
Weight (g)	44	46	47	44	46	43	47	44	48	46	**45.5**

Table 4-5: 70g product weighing results with Mode 1.

- Mode 2:

Times	1	2	3	4	5	6	7	8	9	10	Averager
Speed (rpm)	58	56	54	59	60	59	58	61	62	57	**58.4**
Speed (m/s)	0.079	0.076	0.074	0.080	0.082	0.080	0.079	0.083	0.084	0.078	**0.080**
Weight (g)	40	42	47	42	40	45	46	44	43	45	**43.4**

Table 4-6: 70g product weighing results with Mode 2.

- Mode 3:

Times	1	2	3	4	5	6	7	8	9	10	Averager
Speed (rpm)	69	70	72	74	76	77	75	78	76	76	**74.3**
Speed (m/s)	0.094	0.095	0.098	0.101	0.103	0.105	0.102	0.106	0.103	0.103	**0.101**
Weight (g)	38	41	41	39	46	44	42	42	41	40	**41.4**

Table 4-7: 70g product weighing results with Mode 3.

- Mode 4:

Times	1	2	3	4	5	6	7	8	9	10	Averager
Speed (rpm)	87	88	88	89	90	92	88	91	91	92	**89.6**
Speed (m/s)	0.118	0.120	0.120	0.121	0.123	0.125	0.120	0.124	0.124	0.125	**0.122**
Weight (g)	35	38	31	35	35	37	34	36	38	39	**35.8**

Table 4-8: 70g product weighing results with Mode 4.

4.3.3 Survey weighing 90g products with different speeds

- Mode 1:

Times	1	2	3	4	5	6	7	8	9	10	Averager
Speed (rpm)	46	45	48	47	49	46	45	47	48	46	**46.7**
Speed (m/s)	0.063	0.061	0.065	0.064	0.067	0.063	0.061	0.064	0.065	0.063	**0.064**
Weight (g)	62	68	53	59	59	64	60	61	62	58	**60.6**

Table 4-9: 90g product weighing results with Mode 1.

- Mode 2:

Times	1	2	3	4	5	6	7	8	9	10	Averager
Speed (rpm)	58	62	63	63	57	57	63	62	61	60	**60.6**
Speed (m/s)	0.079	0.084	0.086	0.086	0.078	0.078	0.086	0.084	0.083	0.082	**0.082**
Weight (g)	55	62	52	56	52	61	62	58	60	61	**57.9**

Table 4-10: 90g product weighing results with Mode 2.

❖ Mode 3:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	79	80	79	75	74	75	75	76	76	74	**76.3**
Speed (m/s)	0.108	0.109	0.108	0.102	0.101	0.102	0.102	0.103	0.103	0.101	**0.104**
Weight (g)	55	52	56	51	50	61	55	58	59	60	**55.7**

Table 4-11: 90g product weighing results with Mode 3.

❖ Mode 4:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	92	92	93	91	93	91	90	90	88	89	**90.9**
Speed (m/s)	0.125	0.125	0.127	0.124	0.127	0.124	0.123	0.123	0.120	0.121	**0.124**
Weight (g)	42	53	50	42	47	44	45	47	48	49	**46.7**

Table 4-12: 90g product weighing results with Mode 4.

4.3.4 Survey weighing 100g products with different speeds

❖ Mode 1:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	48	46	49	47	47	50	45	45	44	48	**46.9**
Speed (m/s)	0.065	0.063	0.067	0.064	0.064	0.068	0.061	0.061	0.060	0.065	**0.064**
Weight (g)	73	68	71	74	70	72	69	72	71	68	**70.8**

Table 4-13: 100g product weighing results with Mode 1.

❖ Mode 2:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	66	64	57	64	62	65	60	62	64	63	**62.7**
Speed (m/s)	0.090	0.087	0.078	0.087	0.084	0.088	0.082	0.084	0.087	0.086	**0.085**
Weight (g)	71	64	65	73	73	68	71	67	70	68	**69**

Table 4-14: 100g product weighing results with Mode 2.

❖ Mode 3:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	78	76	77	77	78	78	75	75	74	76	**76.4**
Speed (m/s)	0.106	0.103	0.105	0.105	0.106	0.106	0.102	0.102	0.101	0.103	**0.104**
Weight (g)	62	63	62	65	67	64	60	61	62	63	**62.9**

Table 4-15: 100g product weighing results with Mode 3.

❖ Mode 4:

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	91	90	89	91	91	90	89	91	92	90	**90.4**
Speed (m/s)	0.124	0.123	0.121	0.124	0.124	0.123	0.121	0.124	0.125	0.123	**0.123**
Weight (g)	52	61	52	62	66	63	60	58	59	61	**59.4**

Table 4-16: 100g product weighing results with Mode 4.

4.3.5 Table of average measured results

	50	70	90	100
1	33.7	45.5	60.6	70.8
2	32.5	43.4	57.9	69
3	29.6	41.4	55.7	62.9
4	25.2	35.8	46.7	59.4

Table 4-17: Table of average measured results

4.3.6 Ratio table between measured weight and actual weight

	50	70	90	100	Average
1	0.674	0.650	0.673	0.708	0.676
2	0.65	0.620	0.643	0.690	0.651
3	0.592	0.591	0.619	0.629	0.608
4	0.504	0.511	0.519	0.594	0.532

Table 4-18: Ratio table between measured weight and actual weight

4.4 Proposed formula for the model

For this model, the formula for calculating the actual weight for mode 3 will be:

$$m = \frac{m_m}{0.608}$$

Times	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Averager**
Speed (rpm)	79	80	79	75	74	75	75	76	76	74	**76.3**
Speed (m/s)	0.108	0.109	0.108	0.102	0.101	0.102	0.102	0.103	0.103	0.101	**0.104**
Weight (g)	55	52	56	51	50	61	55	58	59	60	**55.7**
Weight TT(g)	90.461	85.526	92.105	83.882	82.237	100.33	90.461	95.395	97.039	98.684	**91.612**

Table 4-19: 90g product weighing results with Mode 3.

❖ **Relative error**

Relative error (%) = (Measured weight - Actual weight) / Actual weight x 100

In which:

- *Measured weight: The weight value measured by the load cell (unit: kg, gram...)*
- *Actual weight: The true standardized weight of the object being weighed (unit corresponding with Measured weight)*

$$Er = \frac{91.612-90}{90} * 100\% = 1.79\%$$

CHAPTER 5. CONCLUSION

5.1 Advantages

- Good repeatability and stability
- Able to operate continuously for long periods
- Low power consumption
- Real-time monitoring of weight and speed
- Adjustable belt speed with multiple levels
- Accurate weight measurement when standing still
- Simple integration using Arduino and common components

5.2 Disadvantages

- Limited weight range (0-1kg)
- Speed range could be larger (currently 0-90 rpm)
- No wireless connectivity or remote monitoring
- No built-in data storage
- Enclosure is probably lacking for industrial environments
- Vibration can potentially affect the weight stability
- Only displays data on a LCD screen, cannot interface to computer or cloud directly

5.3 Development

- Use load cell with higher max capacity
- Choose motor and encoder that can support higher speeds
- Add wireless modules (WiFi/Bluetooth) for remote connectivity
- Include data logging to SD card module for Arduino
- Design and build robust, possibly water/dust proof metal enclosure
- Add shock absorbers for mechanical dampening

In summary, given the components employed, this is a very good Arduino-based dynamic weighing system, with good performance exhibited for stability and accuracy based on the test results. A few improvements to hardware ruggedness, connection, and storage capacities can significantly increase its applicability for real-world applications.

REFERENCES

[1] (N.d.). Retrieved from https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[2] (N.d.-a). Retrieved from https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[3] Staff, L.E. (2022) *In-depth: Interface L298n DC Motor driver module with Arduino*, *Last Minute Engineers*. Available at: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/ (Accessed: 05 December 2023).

[4] (N.d.-a). Retrieved from https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

[5] (N.d.-a). Retrieved from https://nshopvn.com/blog/nguyen-ly-hoat-dong-cua-cam-bien-can-nang-loadcell-cach-su-dung-cam-bien-voi-arduino-de-lam-mot-can-dien-tu-don-gian/

[6] Thiết bị điện công nghiệp Amazen. (n.d.). Retrieved from https://amazen.com.vn/encoder-la-gi.html

[7] Nguyen, H. (2022) Encoder là gì? Những điều Cần Biết VỀ encoder. Ứng dụng encoder là gì ?, Prosensor. Available at: https://prosensor.vn/encoder-la-gi-nhung-dieu-can-biet-ve-encoder/#Ung_dung_ve_bieu_thi_toc_do (Accessed: 05 December 2023). [8] DegrawSt, &amp; Instructables. (2020). Retrieved from https://www.instructables.com/Arduino-Scale-With-5kg-Load-Cell-and-HX711-Amplifi/

APPENDIX

Code on Arduino.ide software

```
#include <HX711.h>
#include <LiquidCrystal.h>

#define MOTORSPEED 6
#define MOTORA A1
#define MOTORB A2
#define BUTTON A3

const int LOADCELL_DOUT_PIN = 4;
const int LOADCELL_SCK_PIN = 5;

#define PULSEPERROTATE 400

unsigned long PulseCount=0;
unsigned long TimeSave=0;
unsigned long TimeCount=0;
unsigned long Speed;

unsigned long CheckTime=0;

const int rs = 12, en = 11, d4 = 10, d5 = 9, d6 = 8, d7 = 7;
LiquidCrystal LCDDisplay(rs, en, d4, d5, d6, d7);

HX711 MyScale;
```

```
 unsigned int WeightValue;

byte Mode=0;

void Stop()
{
  analogWrite(MOTORSPEED,0);
  digitalWrite(MOTORA,LOW);
  digitalWrite(MOTORB,LOW);
  Mode=0;
}

void Run1()
{
  analogWrite(MOTORSPEED,100);
  digitalWrite(MOTORA,HIGH);
  digitalWrite(MOTORB,LOW);
  Mode=1;
}
void Run2()
{
  analogWrite(MOTORSPEED,150);
  digitalWrite(MOTORA,HIGH);
  digitalWrite(MOTORB,LOW);
  Mode=2;
}
void Run3()
```

```
{
  analogWrite(MOTORSPEED,200);
  digitalWrite(MOTORA,HIGH);
  digitalWrite(MOTORB,LOW);
  Mode=3;
}
void Run4()
{
  analogWrite(MOTORSPEED,255);
  digitalWrite(MOTORA,HIGH);
  digitalWrite(MOTORB,LOW);
  Mode=4;
}



void CountPulse()
{
  PulseCount++;
}

void ScaleStartUp()
{
  Serial.println("Before setting up the scale:");
  Serial.print("read: \t\t");
  Serial.println(MyScale.read());      // print a raw reading from the ADC

  Serial.print("read average: \t\t");
  Serial.println(MyScale.read_average(20));   // print the average of 20 readings from the ADC


  Serial.print("get value: \t\t");
  Serial.println(MyScale.get_value(5));   // print the average of 5 readings from the ADC minus the tare weight (not set yet)


  Serial.print("get units: \t\t");
  Serial.println(MyScale.get_units(5), 1);  // print the average of 5 readings from the ADC minus tare weight (not set) divided
            // by the SCALE parameter (not set yet)


}

void DisplayMain()
{
  LCDDisplay.clear();
  LCDDisplay.print("TD:    v/p   M: ");
  LCDDisplay.setCursor(0,1);
  LCDDisplay.print("CAN NANG:    g");
}

void DisplayMode()
{
  LCDDisplay.setCursor(15,0);
  LCDDisplay.print(Mode);
}
```

```
void DisplaySpeed()
{
  LCDDisplay.setCursor(3,0);
  if(Speed<1000) LCDDisplay.print(" ");
  if(Speed<100) LCDDisplay.print(" ");
  if(Speed<10) LCDDisplay.print(" ");
  LCDDisplay.print(Speed);
}

void DisplayWeight()
{
  LCDDisplay.setCursor(9,1);
  if(WeightValue<1000) LCDDisplay.print(" ");
  if(WeightValue<100) LCDDisplay.print(" ");
  if(WeightValue<10) LCDDisplay.print(" ");
  LCDDisplay.print(WeightValue);
}
void setup() {
  // put your setup code here, to run once:
  Serial.begin (9600);
  pinMode(2, INPUT_PULLUP);
  pinMode(MOTORSPEED,OUTPUT);
  pinMode(MOTORA,OUTPUT);
  pinMode(MOTORB,OUTPUT);
  pinMode(BUTTON,INPUT_PULLUP);

  Stop();
  LCDDisplay.begin(16,2);
  DisplayMain();
  DisplayMode();
  attachInterrupt(0, CountPulse, RISING);

  MyScale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  ScaleStartUp();
  MyScale.set_scale(1700.f);
  MyScale.tare();              // reset the scale to 0

  TimeSave=millis();
  CheckTime=millis();

}

void loop() {
  // put your main code here, to run repeatedly:
   TimeCount=millis()-TimeSave;
   if(TimeCount>2000)
   {
    Speed=PulseCount*1000*60/400/TimeCount;
    Serial.print("TOC DO:");
    Serial.println(Speed);
    PulseCount=0;
    TimeSave=millis();
    DisplaySpeed();
   }
```

```
if(millis()-CheckTime>500)
{
  WeightValue=MyScale.get_units(10);
  if(WeightValue>1000) WeightValue=0;
  Serial.print("Gia tri: ");
  Serial.println(WeightValue);
  CheckTime=millis();
  DisplayWeight();
}

if(digitalRead(BUTTON)==LOW)
{
 delay(100);
 while(digitalRead(BUTTON)==LOW);
 if(Mode==0)
 {
   Mode=1;
   Run1();
 }
 else if(Mode==1)
 {
   Mode=2;
   Run2();
 }
 else if(Mode==2)
 {
      Mode=3;
      Run3();
    }
    else if(Mode==3)
    {
      Mode=4;
      Run4();
    }
    else if(Mode==4)
    {
      Mode=0;
      Stop();
    }
    DisplayMode();
   }
}
```