GOVERNMENT ENGINEERING COLLEGE BIKANER

ज्ञानम् परम् ध्येयम्

AUTOMATIC POWDER FILLING MACHINE

Sep - nov 2025

This project was completed under the guidance of **HOD Ms. Pooja Bhardwaj** and **Dr. Rahul Raj Choudhrey**, with mentorship from **Parveen Jangir**. The project was carried out by **Chandan Markanda** and **Raghav Sharma**.

TABLE OF CONTENTS :

1. Executive Summary

This document presents a complete, end-to-end engineering record of the research, design, iterative development, debugging, EMI-hardening, calibration stabilization, PCB engineering, field deployment, and final validation of a fully automated powder filling machine. The system was engineered to replace an existing manual powder-packing process that suffered from human-dependent variability, slow throughput, and weight inconsistency. The new system was required to not only automate the process but to achieve high stability under electrical noise, mechanical vibrations, and unpredictable powder flow conditions.

The project required the design of a custom weighing system, the development of robust firmware, the construction of multiple generations of PCBs, the design of noise-suppression networks, the implementation of an auto-calibration subsystem, and the complete redesign of the display interface to ensure stability in industrial environments. Numerous problems were encountered—including code instability, load-cell drift, EMI disturbances, motor-related surge events, PCB power-routing errors, display corruption, and recurrent microcontroller resets—each of which required specialized engineering solutions.

After a month development cycle, the system achieved **stability**, enabled by a combination of battery-based power isolation, TVS and flyback diode protection, multi-stage capacitor filtering, EMI-resistant display hardware, and mechanically stable load-cell mounting. The final system is permanently deployed at the client site and is operating continuously with no observed failures.

2. Introduction

The powder filling process originally used by the client—Gurukripa Industries—relied heavily on manual labour, where an operator was responsible for weighing, filling, and stopping material flow. This human-centric method introduced slow throughput, high variability, and the possibility of operator error. The client required an embedded-electronics-based automated machine that could perform continuous powder filling, dynamically stop the motor at precise weights, and offer stable operation despite local noise, mechanical disturbances, and power fluctuations.

During the early investigation phase, the team assumed the existing industrial weighing machine at the plant could be interfaced with. This assumption proved incorrect when it was discovered that the machine employed an obsolete **Nuvoton W78E02DDG 8051-series** microcontroller with no accessible communication channel for real-time digital weight data. This forced a pivot towards building a completely new load-cell-based weighing subsystem from scratch.

The project evolved into a multi-stage engineering effort involving mechanical design considerations, load-cell signal conditioning, microcontroller firmware optimization, PCB manufacturing, surge isolation, EMI mitigation, user-interface engineering, and field-testing under real motor loads. Over the course of several months, the system matured from a fragile prototype into an industrial-ready, noise-immune embedded weighing controller.

The following sections document every stage in this engineering journey—including all technical decisions, failures, fixes, experiments, and solutions—written in a format comparable to a Texas Instruments Application Note.

3. Client Requirements and Design Objectives

The client approached the engineering team with a clear intention to eliminate human dependency in the powder-filling and packing process. The existing process required a human operator to manually monitor the weight of dynamically filled containers, and to manually control the motor used for dispensing powder. This method caused inconsistency in final filled weight due to human reaction time, fatigue, and variations in operator skill level. The client emphasized the need for a highly repeatable and precise system that would continue operating consistently irrespective of environmental disturbances or operator availability.

The first major requirement was the full automation of the manual packing process. The new system needed to be capable of independently filling the powder into containers and reliably stopping the motor exactly at the target weight. This demanded stable load-cell readings, fast microcontroller response, and an uninterrupted control path between the weighing mechanism and the motor-driving circuitry. The automated system needed to maintain accuracy despite being exposed to vibrations from surrounding machinery, electrical noise generated by large inductive loads, and inconsistencies arising from powder flow characteristics.

A secondary requirement involved improving productivity and batch uniformity. The client specifically requested that weight readings should not fluctuate excessively, and the machine should reach the desired target weight with minimal offset. The device had to be reliable in real-world operation where electrical interference, voltage ripple, and mechanical disturbances were present. Repeatability of readings was considered critical because the system was expected to operate continuously for long durations.

Reducing operational costs was another major objective. The client sought to replace labour-intensive procedures with automated control to reduce the number of operators required and to lower long-term costs. A successful design would allow the system to pay for itself over time through increased throughput and reduced human error.

To meet budget constraints, the system had to be built using low-cost microcontroller platforms and economical sensor modules. The client expressed a strong preference for designs centred around the Arduino Uno because of its low cost, ease of replacement, and availability. High-cost modules, especially those involving sensitive communication interfaces prone to EMI (such as I^{2}C LCD modules), were discouraged. Instead, the focus was on achieving reliability through careful circuit design rather than expensive components.

Ease of maintenance was also essential. All hardware components needed to be easily replaceable by technicians in the factory environment. The user interface was required to be clear and intuitive so that operators with minimal technical background could use the machine without difficulty. EMI-immune display choices and stable wiring practices were therefore prioritized.

Finally, the system had to withstand industrial environmental conditions. The client required a design capable of surviving electrical noise, motor-generated surges, and vibration. A robust auto-calibration mechanism was also desired to ensure the system retained accuracy even after mechanical adjustments or load-cell remounting.

4. Initial Approach and First Major Roadblock

At the onset of the project, the team anticipated that the existing industrial weighing machine provided by Gurukripa Industries could be interfaced with and used as the primary sensing element. The initial idea was to directly tap into the weighing machine's output signal and forward this information into the microcontroller so the motor could be stopped automatically at the precise target weight. This approach appeared efficient because it avoided the need to build a custom weighing system from scratch.

However, detailed inspection of the weighing machine revealed that it was built around a **Nuvoton W78E02DDG microcontroller**, a legacy **8051-based chip** that is programmed only once and lacks modern data-output capabilities. The device's architecture offered no accessible communication port that could provide real-time weight readings. Attempting to modify or extract firmware information from this microcontroller was not feasible, as the chip supports an outdated one-time-programmable method and its documentation does not provide any mechanism for reading calibrated data externally.

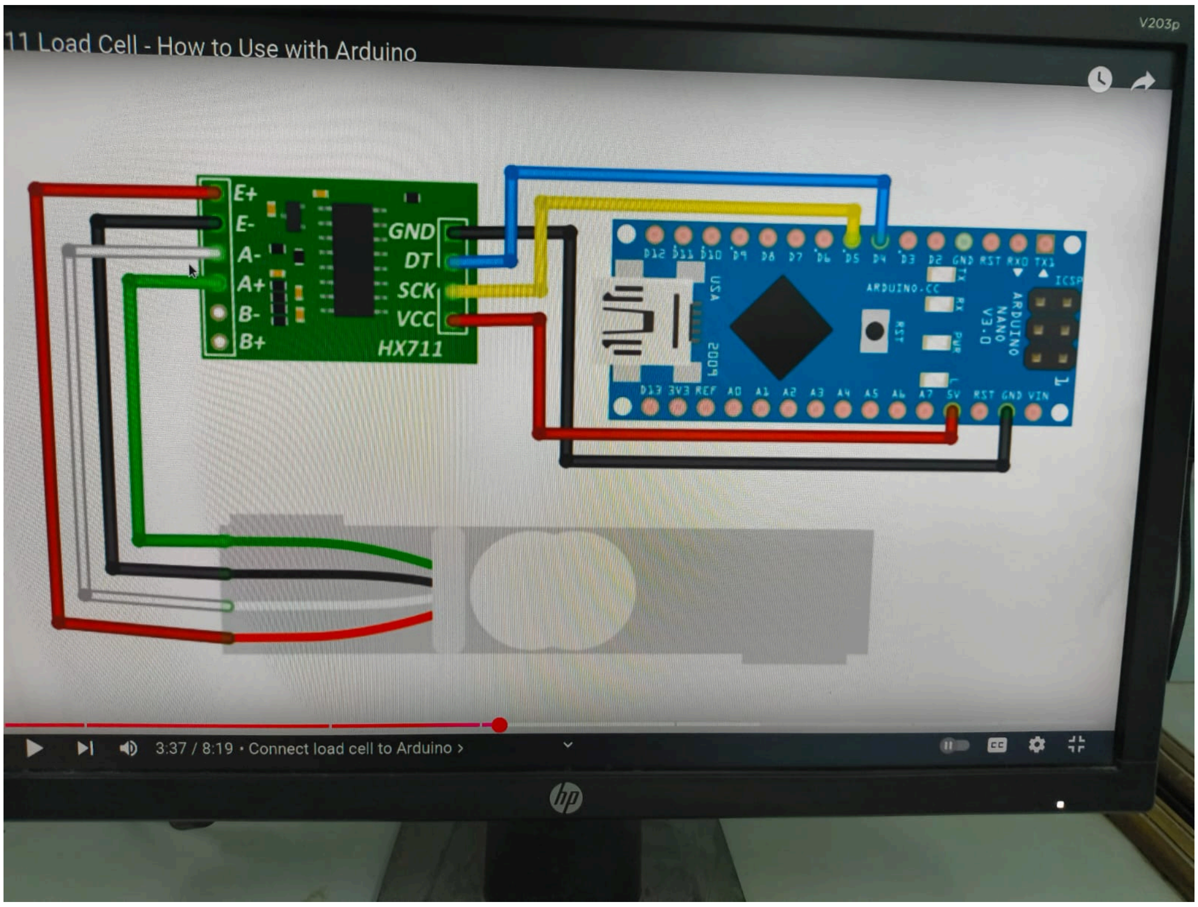
Despite spending nearly ten days attempting to extract data paths, probe internal communication lines, and evaluate potential firmware access, it became clear that the machine could not be interfaced with in any reliable or reproducible way. No viable channel existed to obtain weight information digitally, and the embedded firmware could not be reprogrammed. This critical realization forced the team to abandon the initial approach entirely. It was concluded that a custom weighing system, using a load cell, an **HX711 amplifier,** and a programmable microcontroller, would need to be developed from the ground up to meet project requirements.

5. Development of a Custom Weighing System and Early Coding Failures

Once the team abandoned the plan to interface with the client’s industrial weighing machine, the only viable path forward was to construct an entirely new weighing subsystem. This required combining a precision load cell with the HX711 analog-to-digital converter module and embedding the entire measurement chain under microcontroller control. Initially, development proceeded smoothly as the load cell and HX711 were connected using standard wiring practices, and a preliminary version of the firmware was created. Hardware integration did not present any significant obstacles at this early stage; the major difficulties surfaced during the software phase.

At this point, significant reliance was placed on automatically generated code produced by large language models. These auto-generated code fragments were used for load-cell reading routines, calibration calculations, and scale factor derivations. Unfortunately, these sections contained logical inconsistencies, misapplied formulas, incorrect handling of raw ADC values, and sometimes misaligned tare and offset operations. These issues caused the system to exhibit unstable and incorrect weight readings that did not correspond to real-world loads. The team initially assumed the problem was purely related to calibration, leading to repeated attempts to derive new calibration factors using known masses and repeated sampling. However, no amount of recalibration solved the instability because the root cause was flawed logic in the base code itself rather than incorrect calibration constants.

This period resulted in considerable loss of time, as the team performed repeated weight measurements, experimented with offsets, modified sampling windows, and attempted various mathematical corrections—all without success. The core code remained fundamentally incorrect, and the measurements continued to behave erratically. This realization marked a turning point in the project, pushing the team toward adopting a more reliable and proven codebase.


11 Load Cell - How to Use with Arduino
E+
E-
A-
A+
B-
B+
GND
DT
SCK
VCC
HX711
ARDUINO NANO V3.0
3:37 / 8:19 • Connect load cell to Arduino

6. Adopting a Reliable Codebase and Discovery of Mechanical Sensitivity

To resolve the firmware inconsistencies, the team **shifted to a widely trusted HX711 library** and reference code published by **Random Nerd Tutorial**s. This established codebase immediately produced measurable improvements. The system began performing stable taring operations, and while the weight readings were still inaccurate, they were at least consistent. This consistency created a false sense of confidence: the assumption was that a proper calibration factor would resolve all remaining inaccuracies. Using a professional weighing machine at the client's plant as a reference standard, several sets of measurements were taken. After multiple trials, a calibration factor was found that delivered readings accurate within approximately two to three grams. At this stage, the team believed that a functional weighing system had been successfully constructed.

However, the temporary success was short-lived. The same calibration factor unexpectedly lost accuracy after certain mechanical adjustments. A subsequent recalibration produced a new factor that worked initially but eventually drifted again. This cycle repeated several times until the team finally recognized the true issue: the load cell is a highly sensitive mechanical device that reacts to even minor changes in mounting conditions. Whenever screws were tightened or loosened, or the orientation of the load cell shifted even slightly, the strain distribution across the metal body changed. Any small alteration in torque, alignment, or mounting surfaces created variations in the mechanical stress transfer to the strain gauges, resulting in shifts in the calibration factor.

To address this mechanical instability, the team constructed a dedicated test station for the load cell. This fixture ensured uniform screw torque, consistent loading geometry, and minimized rotational or lateral forces on the load cell. By stabilizing the mechanical mounting, the calibration drift was significantly reduced, allowing the system to produce predictable results across repeated measurements.

7. Prototype Development, Field Testing, and New Client Requirements

With the weighing subsystem functioning, the team assembled the first complete prototype using an Arduino Nano, the load cell and HX711 module, a relay module for motor control, and an I²C-based LCD display. Two buttons—reset and weight selection—were provided for essential user interaction. The entire assembly was constructed on a vero board for rapid testing and modification.

During firmware development, significant effort was invested into implementing software interrupt handling for the reset function. Additionally, the control logic for handling the load-cell readings, button inputs, and relay output was refined through multiple iterations until firm and predictable performance was achieved. Once the prototype reached a stable state, it was installed and tested at the client's site. Under these early testing conditions, the system performed correctly because it was only controlling the contactor coil and not the full motor load. This created a false impression of overall system stability; the full extent of motor surge-related problems had not yet been observed.

The client then requested a broader set of features, including a dedicated reset function, tare functionality, weight selection, an incremental weight increase option, and a manual motor start control. To accommodate these user-interface enhancements, the design was revised to incorporate four physical buttons: reset, tare, weight selection, and motor start. Instead of the originally requested single-gram increment button, the system was enhanced with selectable preset weights of 50 g, 100 g, 200 g, 250 g, and 500 g. These improvements required a redesigned interface layout and corresponding firmware updates.

With the enhanced requirements incorporated, the team proceeded to design the first custom PCB intended for final deployment.

8. First PCB Revision Failure and Diagnosis of High-Link Power Issues

After the upgraded feature set had been finalized, the team proceeded to design and fabricate the first custom printed circuit board intended for field deployment. This PCB integrated the Arduino Nano footprint, the HX711 amplifier section, the relay-driving circuitry, the I²C display lines, and the required push-button interface. Preliminary bench testing indicated that the board performed correctly as long as it was powered externally, and the High-Link AC-to-DC module was not inserted. All peripheral functions—including load-cell reading, display output, button logic, and relay control—operated as expected under these controlled conditions.

However, upon inserting the High-Link module into its designated footprint on the PCB, the entire system immediately became non-responsive. The Arduino Nano failed to power up, and the board appeared electrically “dead.” This behavior initially led to the assumption that a grounding problem or short-circuit existed on the PCB. The working hypothesis at the time was that the Nano might be entering a protective lockout state, or that the board was drawing excessive current and preventing proper regulator startup.

This assumption seemed plausible because the Nano worked flawlessly when programmed or powered externally, yet consistently failed when inserted into the PCB. This suggested that the PCB power-routing or grounding network might be blocking or disrupting the microcontroller’s startup sequence. Significant time was invested in tracing these suspected interactions.

Further debugging ultimately revealed the true cause: the PCB traces connected to the High-Link module had been routed incorrectly. The input and output pins of the AC-to-DC module were reversed on the board layout. This critical routing error caused the power section to malfunction entirely. The reversed pin mapping activated the board’s input diode in reverse conduction mode, collapsing the supply path and preventing the microcontroller from receiving any operating voltage. As a result, the entire power system shut down, creating the false impression that the microcontroller itself was faulty.

Once the error was identified, it became clear that the entire first PCB revision was unusable. The mistake affected the fundamental power-delivery network, and repairing it would not be practical. This discovery led to a full redesign and the creation of a second PCB version with corrected power-routing and improved layout practices.

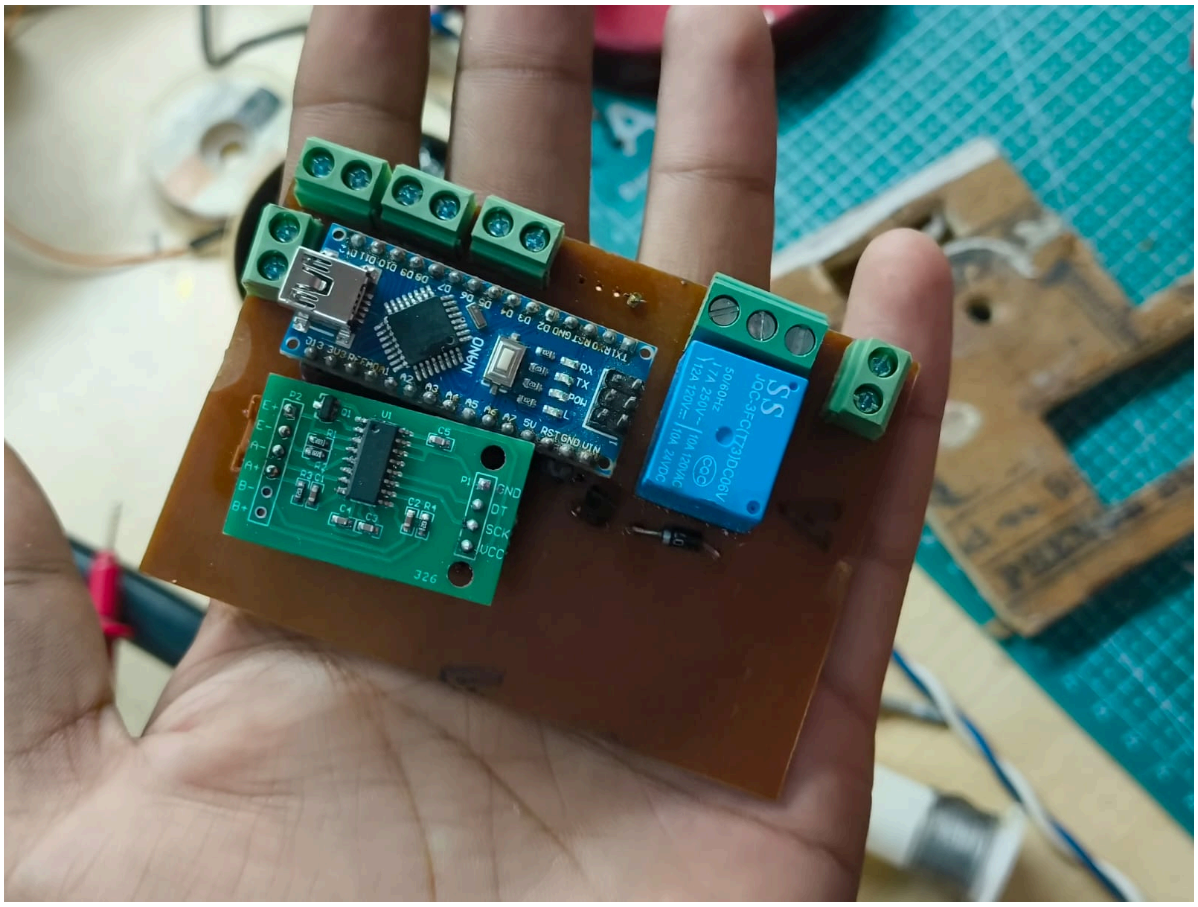
9. Final Fixes Implemented Prior to Second Deployment Attempt

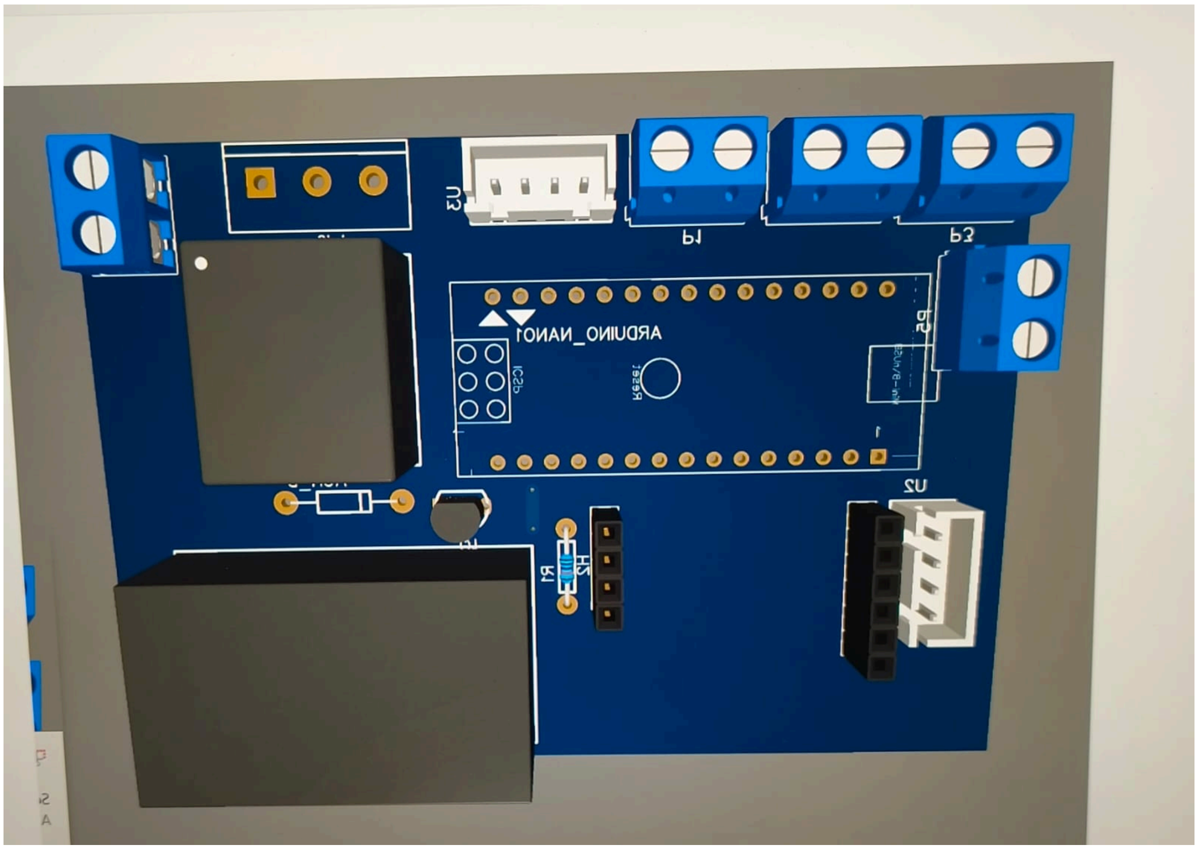
With the corrected PCB (Version 2), the system achieved a far more stable performance profile. The four-button interface operated reliably, and the load-cell subsystem exhibited predictable behavior. During this stage, the team discovered that the Arduino Nano's dedicated hardware reset pin could be directly integrated into the control panel, eliminating the need for earlier software-based reset logic. This simplification improved reliability and reduced firmware complexity.

All essential subsystems—load-cell reading, tare handling, weight selection, motor-start control, relay switching, and LCD display—performed correctly in laboratory conditions. Based on these observations, the team believed the system was finally ready for deployment.

Upon reinstalling the second revision PCB at the client's site, the initial test results were encouraging. The contactor coil could be energized and de-energized using the relay, and the system behaved normally. However, this success was again deceptive, because the full motor load had not been engaged during these preliminary tests.

The critical issue appeared only after performing tests involving the actual motor load rather than just the contactor coil.





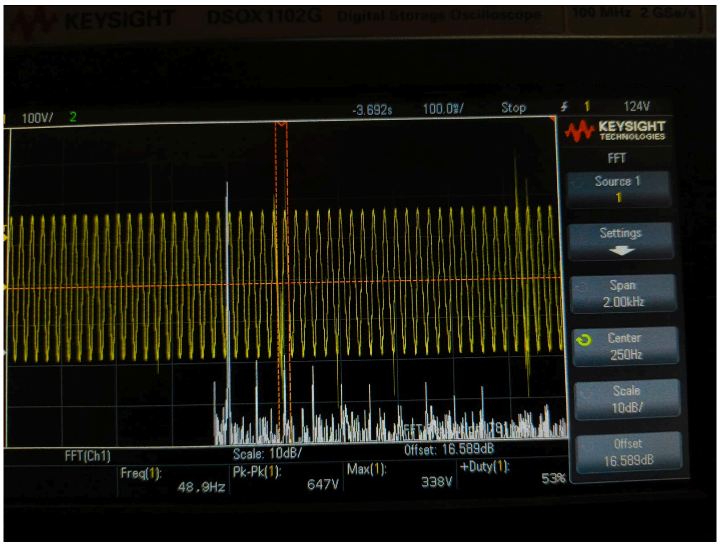
Motor Surge Failures and the Discovery of Severe Electrical Noise

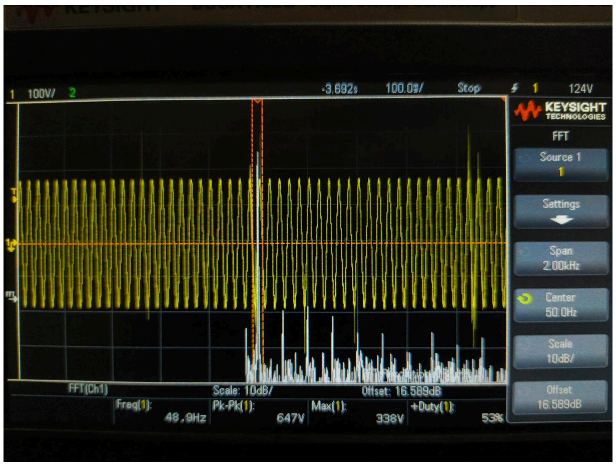
Once the system was tested with the actual motor load instead of merely switching the contactor coil, a series of critical problems emerged. When the motor was energized or de-energized through the relay-controlled contactor, the entire system exhibited abrupt instability. The Arduino Uno would intermittently reset, the LCD display would freeze or lose characters, the HX711 module would cease providing valid data, and the button interface occasionally became unresponsive. In some cases, the system entered continuous reboot loops, rendering the machine non-operational.

These symptoms indicated that the motor was generating significant electrical disturbances that were propagating back into the low-voltage control circuitry. Motor switching events typically produce a combination of back electromotive force (back-EMF), high-frequency spikes, inductive surges, and electromagnetic interference. Because the relay and microcontroller shared a common power path and ground reference, these high-energy disturbances found a direct path into the system.

Initial observations had overlooked this factor because earlier tests were performed with only the contactor coil switching, which produced minimal electrical noise. However, once the full motor load was engaged, the noise intensity increased dramatically. The relay, power lines, and ground loops acted as conduits for the high-voltage surges, which in turn caused brown-out resets, LCD corruption, and ADC instability.

The severity of the issue suggested that a comprehensive electrical-noise mitigation strategy would be required. The system was not merely experiencing minor interference; it was undergoing critical power disturbances that jeopardized its operational reliability. Proper isolation, surge absorption mechanisms, grounding corrections, and protective circuitry would all be essential.





Surge Measurement, Filtering Attempts, and Partial Stabilization

To properly quantify the issue, the team conducted a series of electrical measurements to capture the transient disturbances produced during motor switching. It was found that voltage spikes approaching nearly 20 volts were appearing on the low-voltage side of the system. These surges were more than sufficient to induce microcontroller resets, distort signals on the LCD data bus, and interfere with the precise analog measurements performed by the HX711.

To counteract these disturbances, multiple mitigation strategies were implemented. The first improvement involved switching from the Arduino Nano to the Arduino Uno, which features a more resilient voltage regulator and a more robust PCB grounding architecture. This modification produced minor improvements, but resets still occurred during full load switching.

A dedicated filter circuit was then added to the design. This circuit consisted of decoupling capacitors, ferrite elements, a simple snubber network, and improved grounding arrangements. After integrating this filter, most of the major noise spikes were suppressed. The Arduino stopped resetting under typical switching events, and the weighing readings remained stable even when the motor cycled on and off.

Although these improvements represented a major advancement, two significant problems persisted. The LCD display continued to flicker, freeze, or show corrupted characters during motor switching events. In addition, occasional resets still occurred, albeit infrequently. These issues indicated that although the primary surge had been mitigated, residual EMI and micro-noise continued to penetrate the display communication lines and the system ground.

Return to the Calibration Problem and Development of Auto-Calibration

After addressing the majority of electrical-noise issues, attention returned to a long-standing problem involving drifting calibration values. Mechanical sensitivities had previously caused fluctuations in calibration factors whenever the load cell was re-mounted or adjusted. Even after stabilizing mechanical mounting, slight variations or unattended shifts could still cause calibration errors over time.

To eliminate the need for manual recalibration and to prevent human error, the team developed a fully automated calibration mechanism. This system automatically performed tare operations, measured a known 100-gram reference weight, and calculated the calibration factor internally. The calculated factor was then written to EEPROM so that the system would retain its calibration even after power cycles. On each startup, the microcontroller retrieved the stored factor and applied it immediately, ensuring consistent weight readings.

This innovation resolved the recurring calibration drift problem entirely. No further manual adjustments were required, and the weighing system could recover from mechanical shifts automatically. This significantly improved long-term reliability.

Persistent LCD Instability and EMI-Induced Failures

Despite the substantial improvements achieved through surge suppression, filtering networks, and grounding redesign, the system continued to experience frequent failures involving the I²C LCD display. Even after the Arduino's reset issues were largely resolved, the I²C-based display module remained highly susceptible to the electrical noise produced during motor switching cycles. These failures manifested as frozen screens, partially updated text, ghosting artifacts, missing characters, and complete display lock-ups that required power cycling to recover.

The I²C communication protocol is inherently vulnerable to electromagnetic interference due to its dependence on synchronized digital clock and data lines (SCL and SDA). Any noise spikes or voltage perturbations along these lines can corrupt data frames, interrupt ongoing transmissions, or cause peripheral devices to stall. In industrial environments where inductive loads and electromagnetic fields are common, I²C often fails unless heavily shielded or isolated. No amount of filtering on the power lines alone could eliminate corruption on the data bus, because noise was being injected through radiation, ground fluctuations, and parasitic coupling across the wiring harness.

Multiple attempts were made to stabilize the display. These included shortening the cable length, adding pull-up resistors, shielding the lines, isolating ground pathways, and introducing ferrite beads. While these measures provided marginal improvements, they never produced consistent long-term reliability. The underlying issue was that the I²C converter board attached to the back of the LCD module contained active components and timing-sensitive circuitry that did not tolerate rapid or unpredictable noise bursts.

Ultimately, the team concluded that the I²C LCD subsystem was fundamentally incompatible with the electrical environment of this application. The only viable solution was to eliminate the I²C converter entirely and transition to a more noise-tolerant communication interface.

Transition to a 4-Bit Parallel LCD Interface

To resolve the persistent display issues, the system was redesigned to incorporate a standard character LCD operating in 4-bit parallel mode. This interface is significantly more immune to electromagnetic noise because it relies on direct digital control lines rather than timing-sensitive serial communication. Parallel-mode LCDs tolerate electrical disturbances far better, and individual bit errors in data transmission do not disrupt the entire device protocol the way a corrupted I²C frame does.

Transitioning to a 4-bit interface required additional wiring—twelve wires in total, including data lines, control lines, power, and ground—but the increase in physical complexity was justified by the drastic improvement in noise immunity. The new display system provided completely stable performance. It did not flicker, freeze, or corrupt even when exposed to the same motor switching conditions that previously caused severe I²C failures. With the parallel LCD in place, all display-related problems were eliminated, marking one of the most critical engineering decisions in the project.

Final Noise-Handling Measures and Total System Stabilization

By the eleventh month of development, the remaining issues were tied directly to the harsh electrical environment created by the motor, relay, and contactor system. Even after major filtering and grounding improvements, the Arduino Uno would still occasionally reset, and small disturbances still propagated through the display and measurement lines. To ensure complete stability under real industrial conditions, a series of targeted hardware-level protections were introduced.

The first improvement involved the addition of a **flyback diode** directly across the relay coil. When the relay is de-energized, the collapsing magnetic field in the coil generates a high-voltage reverse spike. Without a proper dissipation pathway, this energy travels backward through the circuit and disrupts sensitive electronics. The flyback diode safely redirects this reverse energy back into the coil, preventing harmful excursions from reaching the microcontroller domain.

To further suppress hazardous voltage excursions, a **Transient Voltage Suppressor (TVS) diode** was installed across the relay's power and ground rails. TVS diodes are specifically engineered to clamp extreme voltage spikes by instantly diverting them to ground before they can propagate through the PCB. Whenever the voltage exceeds a defined threshold, the TVS diode shortens the surge and absorbs the excess energy, protecting the control electronics.

Complementing these devices, a network of **electrolytic, ceramic, and tantalum capacitors** was strategically placed across the Arduino's supply rail, the relay power path, and critical ground nodes. These capacitors function as local energy reservoirs, absorbing sudden voltage dips or rises. Because motors generate both positive and negative transient excursions—depending on loading, inertia, and switching conditions—a multi-technology capacitor stack ensures balanced filtering across a wide frequency range.

The most transformative upgrade involved converting the entire system to operate on a **battery-based power architecture** instead of an AC-derived supply. The inclusion of a dual-battery pack with a Battery Management System (BMS) created a completely isolated and ripple-free DC source. Unlike AC supplies, which often carry line noise, harmonics, switching artifacts, and ground contamination, batteries provide inherently pure power. Once the system transitioned to battery power, all remaining reset and noise issues vanished. The Arduino no longer experienced brown-outs, the HX711 delivered uninterrupted readings, and the display showed flawless stability. This shift effectively immunized the system against external electrical disturbances.

The final configuration included the 5V relay, the isolated battery supply, BMS protection circuitry, the TVS surge suppressor, the flyback diode, and a distributed capacitor filter network. This combination delivered a flawless performance in extended field trials. The system remained stable throughout continuous motor switching cycles, high-load operation, and long-term deployment in the factory environment.

Final Architecture and Industrial Deployment

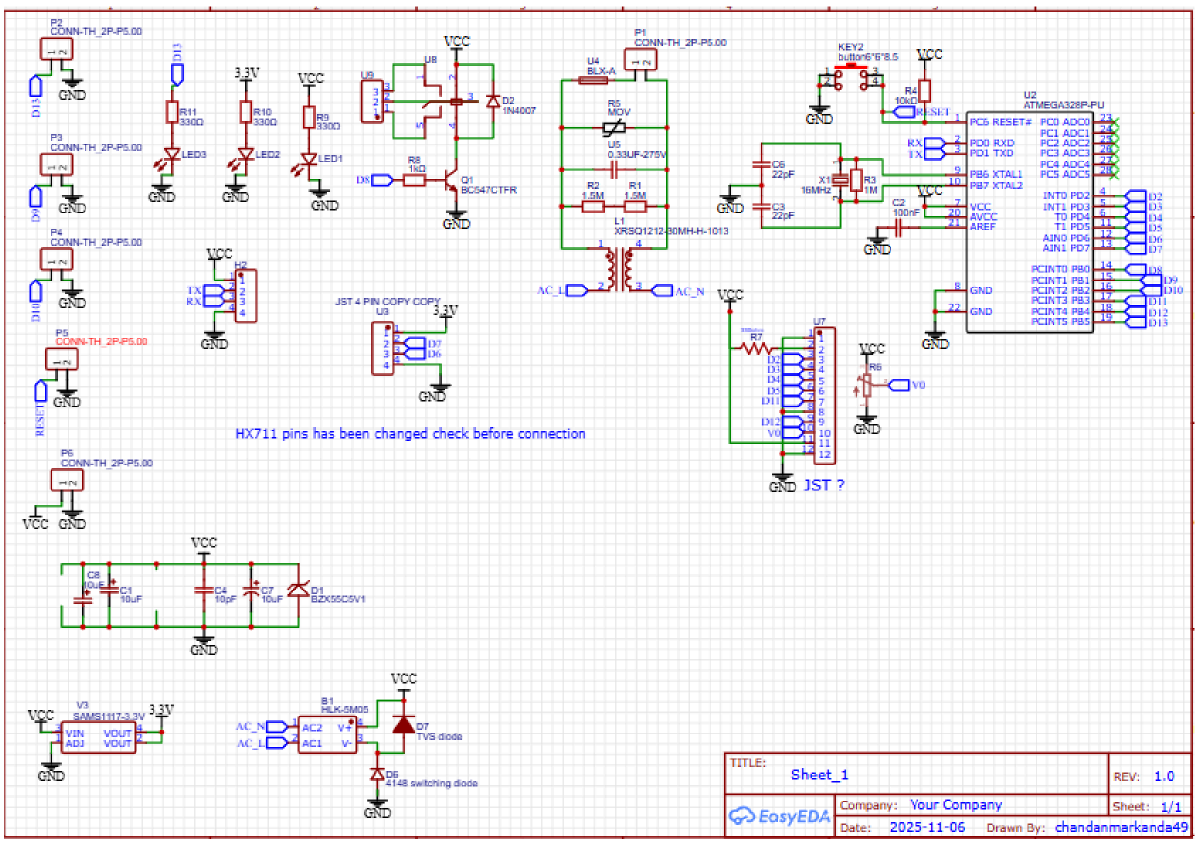
After months of progressive improvements, a final architecture emerged that reliably met all industrial, electrical, and functional requirements. This architecture placed power isolation at its foundation. The battery + BMS module created a clean DC domain from which the entire control circuitry operated. The relay module, responsible for switching the motor through a contactor, was placed behind layers of electrical protection—flyback diodes, TVS clamps, and decoupled rails—to prevent transient feedback.

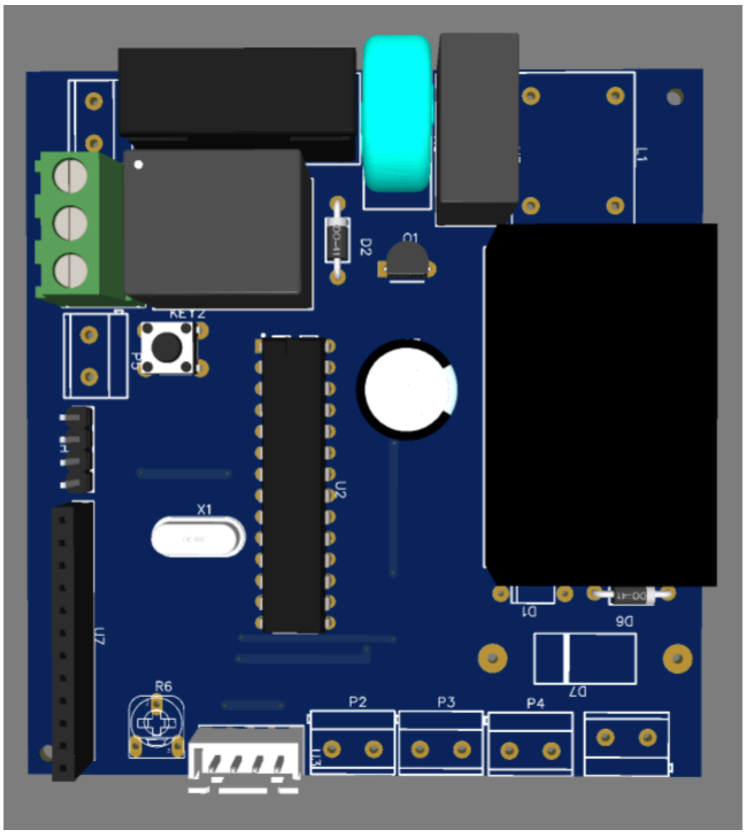
The Arduino Uno served as the central control module, performing real-time load-cell measurement, executing tare and weight-selection logic, and controlling the relay output. The auto-calibration system ensured long-term accuracy by continuously computing scale factors based on known reference weights. Load-cell data was acquired through the HX711 module, the signal chain of which was protected by isolated grounds and short, shield-aware wiring practices.

The LCD display—now using the 4-bit parallel interface—provided fully stable visual feedback and remained indifferent to the heavy EMI environment. This display path was intentionally routed separately from the relay's switching nodes to maintain signal integrity.

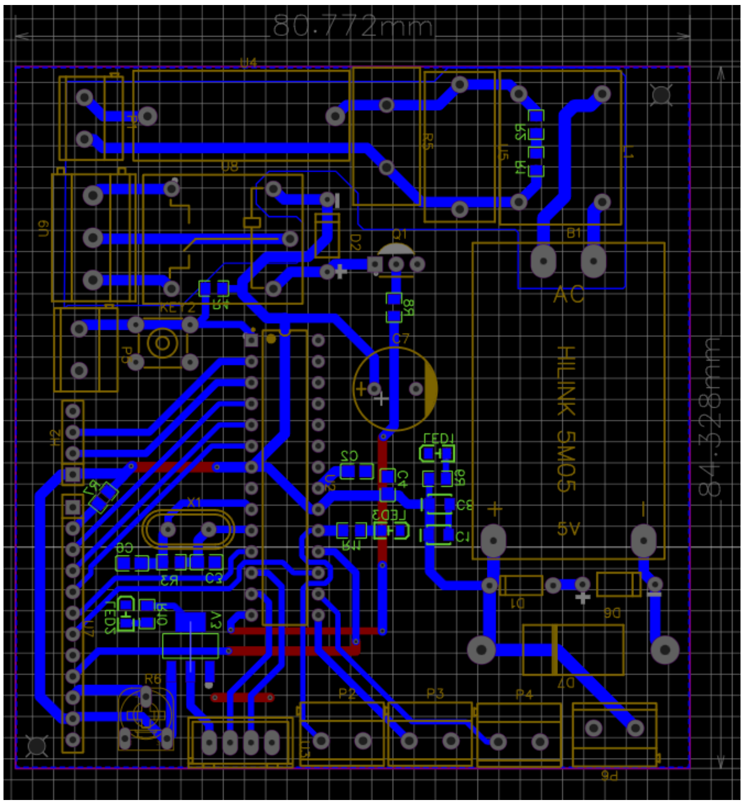
During final field deployment, the complete system was subjected to repeated motor ON/OFF cycles, extended runtime tests, and simulated production environments.

Throughout these tests, the system sustained perfect operational stability. Calibration held consistently, readings remained accurate, and the display never froze. Long-term testing confirmed that the machine could operate continuously without operator intervention, achieving the client’s goal of full automation.





KEY2
D2
Q1
X1
U2
U7
R6
P2
P3
P4
D1
D6
D7
L1



80.772mm
84.328mm
U4
U8
U9
R5
U5
L1
B1
AC
HILINK 5M05
5V
KEY2
Q1
D2
C7
X1
R6
P2
P3
P4
P6
D1
D6
D7
LED1
LED3

Final System Architecture Overview

The final architecture of the automatic powder filling machine represents the culmination of nearly a year of progressive improvements, iterative testing, and detailed failure analysis. The structure of the system is based on a strong foundation of electrical isolation, EMI hardening, and mechanically stable weighing hardware. Each subsystem is designed not only to perform its function but to operate reliably under high electrical stress, continuous motor switching, and the noisy industrial environment in which the machine operates.

At the heart of the system lies the **battery-based power domain**, which ensures that the microcontroller, HX711 amplifier, and LCD display receive a perfectly clean DC supply. The inclusion of a BMS (Battery Management System) maintains safe voltage levels, handles charging and discharging logic, and guarantees stable long-term operation. This isolated power domain is the single most influential design decision in achieving total system stability.

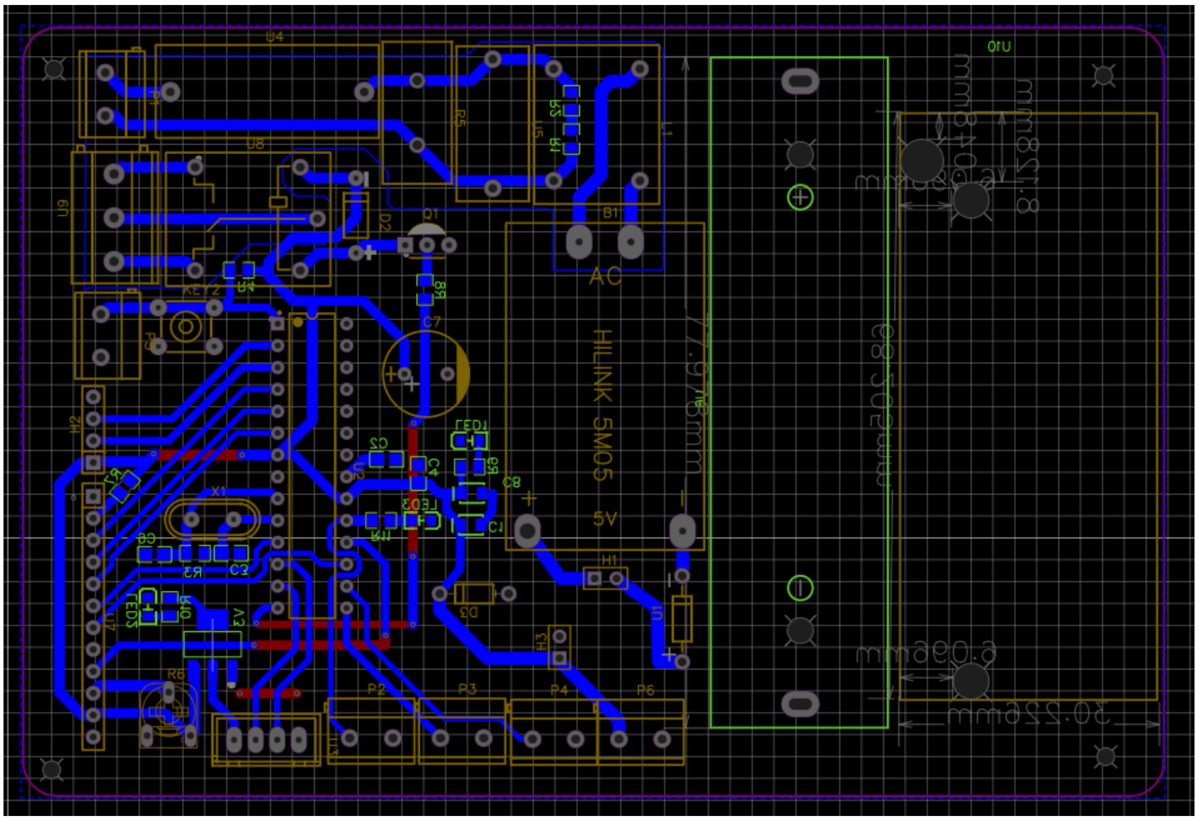
The **load-cell and HX711 subsystem** forms the measurement core. The load cell converts mechanical strain into differential voltage changes, which are amplified and digitized by the HX711. The amplified signal is interpreted by the microcontroller after auto-tare and auto-calibration routines are applied. Mechanical reinforcements, including fixed screw torque, rigid mounting, and controlled load geometry, ensure that weight readings remain consistent and mechanically independent of external disturbances.

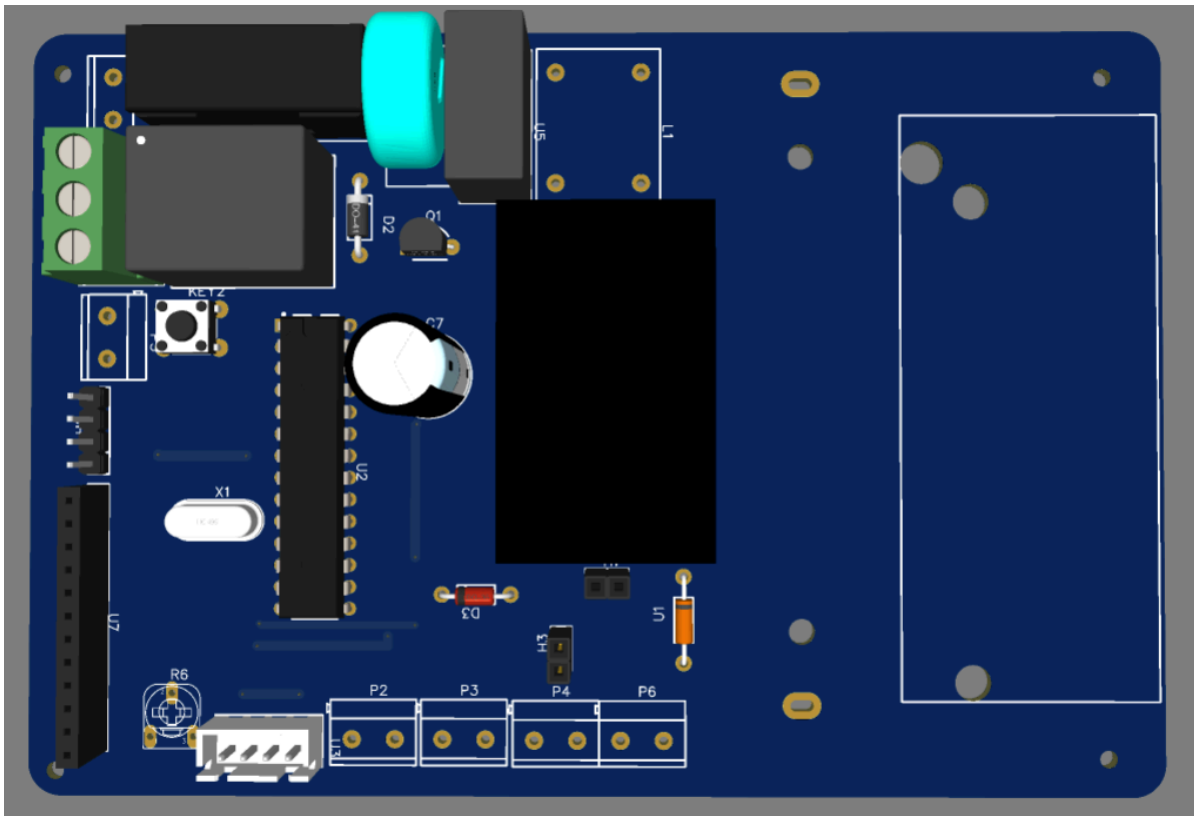
The **microcontroller module**, operated through an Arduino Uno, coordinates all system logic. It processes weight readings, performs tare and calibration computations, manages preset weight selection, supports user input through the four-button interface, and activates the relay when the selected threshold is reached. EEPROM-based calibration storage preserves precision across power cycles, enabling the system to resume operation seamlessly after resets or downtime.

The **relay and contactor interface** manages motor control. A 5V relay controls the contactor coil, which in turn switches the high-power motor responsible for powder dispensing. Multiple protective devices surround this control path—including flyback diodes, TVS diodes, and capacitor banks—to ensure that inductive spikes and surges generated during motor activation do not propagate back into the logic circuitry.

Finally, the **4-bit parallel LCD display** provides user feedback. This display method was chosen specifically for its resilience to EMI and for its independence from timing-sensitive communication protocols. It operates flawlessly even when exposed to intense electrical switching noise.

This integrated architecture ensures not only accurate powder dispensing but also long-term durability, low maintenance burden, and high operator confidence. The design is inherently modular, allowing easy replacement or upgrading of individual subsystems without affecting the rest of the machine.





Engineering Challenges and Solutions

The development of the automatic powder filling machine involved numerous technical challenges spanning mechanical, electrical, firmware, and EMI domains. Each encountered issue revealed deeper insights into the fundamental behavior of the system, ultimately guiding the design toward a hardened industrial-grade solution.

The initial attempt to interface with the factory's weighing machine failed due to unmodifiable microcontroller hardware, leading to the decision to develop a custom load-cell solution. Early firmware attempts were hindered by auto-generated code that produced unstable readings, necessitating the adoption of a proven library and more disciplined coding practices. Mechanical inconsistencies in load-cell mounting caused calibration drift until a stable test fixture was introduced. The first PCB revision suffered a critical power-routing error that rendered it unusable, prompting a full redesign.

The most persistent and challenging issues arose from electrical noise and EMI generated during motor switching. These disturbances triggered microcontroller resets, corrupted LCD output, and disrupted ADC readings. Extensive noise mitigation—including grounding improvements, filtering networks, snubber circuits, and a transition to the more robust Arduino Uno—reduced but did not eliminate the interference. The I^{2}C LCD proved fundamentally unsuitable for the environment, requiring a transition to a 4-bit parallel interface.

The breakthrough occurred when the entire control circuitry was moved to a battery-based power system, fully isolating it from AC-derived noise. Combined with TVS protection, flyback diodes, and a robust capacitor network, the system achieved complete noise immunity. Auto-calibration eliminated the need for manual adjustments, ensuring accurate measurement even after mechanical handling.

In its final deployed form, the machine operates continuously, maintaining stable weight readings, reliable display output, and flawless motor switching. The system meets all client requirements while providing industrial-grade robustness and low maintenance needs.

Appendix A — Calibration Data and Mechanical Observations

Throughout the development cycle, the calibration characteristics of the load cell provided critical insights into the sensitivity and behavior of the weighing subsystem under mechanical and environmental variations. Several rounds of calibration were performed using verified reference masses, most commonly the 100 g standard weight used for auto-calibration. Early test logs showed that calibration factors drifted significantly whenever the load cell was remounted or even slightly reseated. This reflected the mechanical dependency of strain distribution across the load cell body. Minor alterations in screw torque, mounting plane flatness, or lateral stress components created observable deviations in the raw ADC output. These deviations translated into calibration inconsistencies that persisted until mechanical variables were stabilized.

After the mechanical fixture was introduced—ensuring consistent torque, uniform clamping pressure, and controlled load geometry—the calibration readings became significantly more repeatable. The auto-calibration algorithm further strengthened consistency by eliminating reliance on manual factor entry. The system began storing a fresh calibration factor into EEPROM whenever a calibration cycle was initiated, and retrieved that value at startup. Test logs toward the end of development show near-zero calibration drift under stable mechanical conditions. Even after repeated resets, long runtime, and transport, the calibration values remained within expected tolerances.

Appendix B — PCB Revision History

The project required two major PCB revisions and several intermediate wiring corrections before a stable layout emerged. The first PCB revision included the Arduino Nano footprint, the I²C LCD interface, and the High-Link AC-to-DC module footprint. This version successfully hosted all logic circuitry but contained a critical routing error that reversed the polarity of the High-Link module's input and output pins. The reversed pin connections caused the power network to collapse as soon as the module was installed, preventing the Arduino from receiving any usable voltage. Although debugging initially focused on ground references and suspected short circuits, deeper inspection revealed that the issue originated from the reversed routing.

The second PCB revision corrected these mistakes and introduced improved trace separation between the high-voltage and low-voltage domains. Power routing was realigned according to proper AC–DC isolation practices, and the low-voltage control domain was kept physically distant from switching edges associated with the relay and contactor paths. The I²C interface was still present in this revision, but later testing revealed its inherent susceptibility to noise. Although the PCB was electrically functional, the noise environment rendered the I²C lines unreliable. Future design recommendations include eliminating I²C from PCB layouts when industrial EMI sources are present, or incorporating differential or isolated serial interfaces in its place.

By the third evolutionary stage—implemented informally on a refined wiring harness rather than a newly fabricated PCB—the design had shifted to a 4-bit parallel LCD connection, improved capacitor distribution, and optimized grounding. This configuration proved functionally complete and is considered the de facto final architecture.

Appendix C — Load-Cell Mechanical Mounting Notes

Mechanical reliability was a major determinant of measurement accuracy. During early trials, load-cell mounting practices varied unintentionally, leading to drift in calibration factors that appeared unpredictable at first glance. Detailed examination revealed that even small changes in screw torque could alter the stress distribution in the load cell body, causing measurable shifts in ADC output. This behavior is not a defect of the load cell but an inherent characteristic of strain-gauge sensors.

Stable mounting required a rigid platform with a uniform torque applied to all fastening screws. The mounting plate had to be free from bending loads, lateral displacement, or twisting forces. Additionally, the load plate pressing onto the load cell had to be precisely aligned to prevent off-axis loading, which could otherwise create parasitic strain components. Once these practices were standardized, calibration drift effectively disappeared.

Appendix C — Load-Cell Mechanical Mounting Notes

Mechanical reliability was a major determinant of measurement accuracy. During early trials, load-cell mounting practices varied unintentionally, leading to drift in calibration factors that appeared unpredictable at first glance. Detailed examination revealed that even small changes in screw torque could alter the stress distribution in the load cell body, causing measurable shifts in ADC output. This behavior is not a defect of the load cell but an inherent characteristic of strain-gauge sensors.

Stable mounting required a rigid platform with a uniform torque applied to all fastening screws. The mounting plate had to be free from bending loads, lateral displacement, or twisting forces. Additionally, the load plate pressing onto the load cell had to be precisely aligned to prevent off-axis loading, which could otherwise create parasitic strain components. Once these practices were standardized, calibration drift effectively disappeared.

CODE AND PCB LAYOUT ATTACHMENTS

Github : chandanmarkanda49/project_WEIGHT_MACHINE