The Codefathers |
17/11/2025

THIRD EVALUATION

FEEDBACK BASED SPEED CONTROL OF DC MOTOR

NAME OF PROJECT:
FEEDBACK BASED SPEED AND POSITIONAL CONTROL OF DC MOTOR

PRESENTED BY:
PARTH
SAHAJ
KAVISH
DHYEY

PRESENTED TO:
PROFESSOR SACHIN CHAUDHARI
PROFESSOR AFTAB HUSSAIN
PROFESSOR DEEPAK GANGADHAR

Index

Problem Statement

Speed and positional control of a DC motor using a RPM sensor.

01 Speed control of D.C. motor by a linear PWM relation is not possible, as the relationship b/w RPM and PWM of a D.C. motor is non-linear in nature.

02 The non-linear relationship implies that we need to implement a PID control that makes the motor fast and responsive.

03 Handle overshooting and slow response times that are caused by a linear PID controller.

Motivation

Our motivation for doing this project is:

Motivation #1

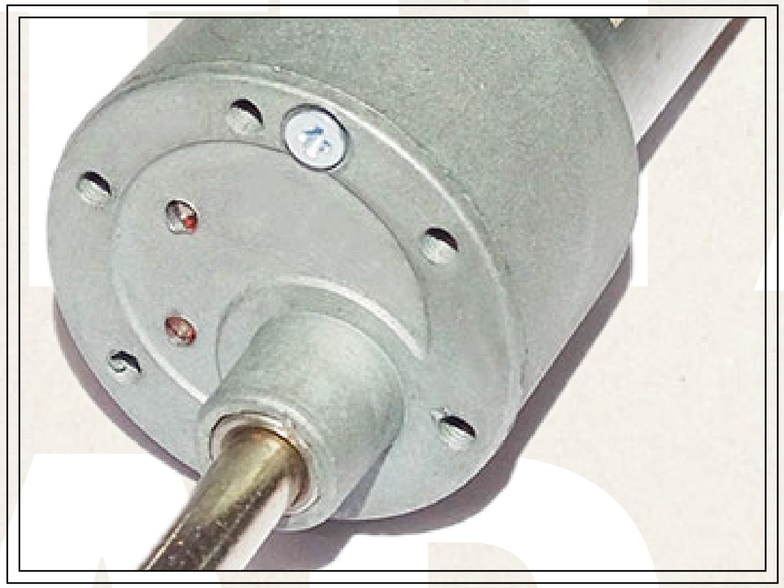
Overcoming the non-linearity of relationship b/w RPM and PWM for fast and stable control.

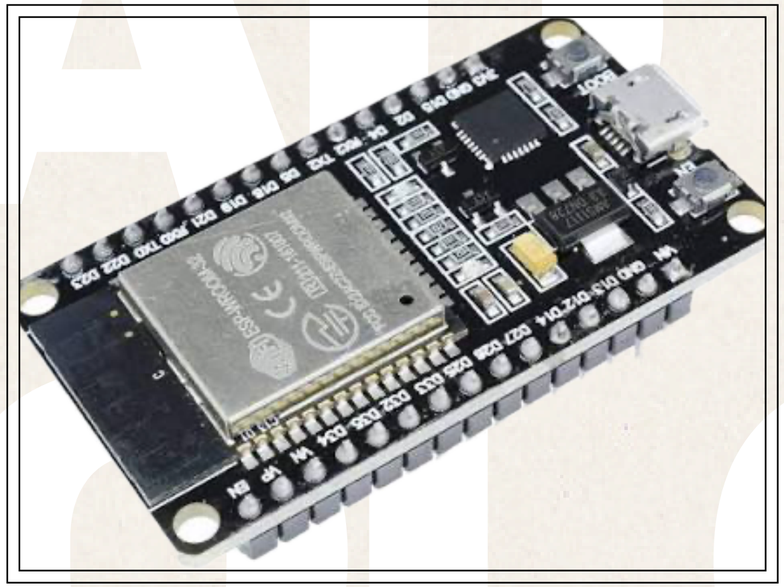
Motivation # 2

Achieving a robust system for advanced positional control of a DC motor.

Motivation # 3

Enabling precise, quick and stable operations of a DC motor

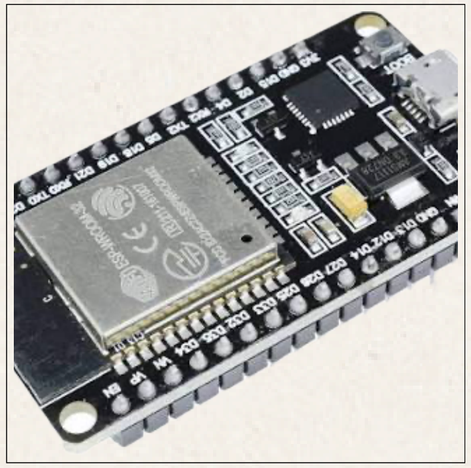




Methodology

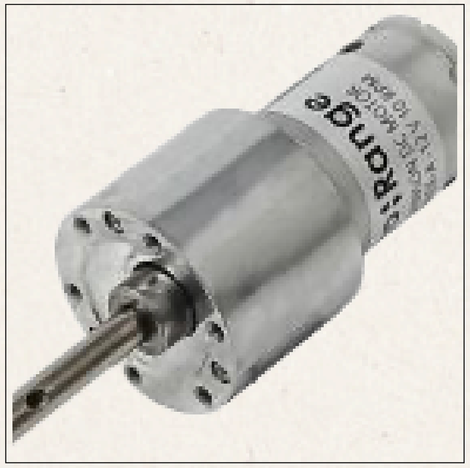
Parameters, Sensors & MCUs

Parameters, Sensors & MCUs



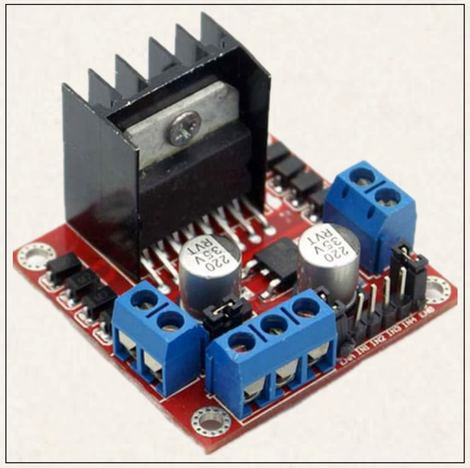
Microcontroller Unit

ESP-32 Dev Module



D.C. Motor

Johnson (12 V & 300 RPM)



Motor Driver

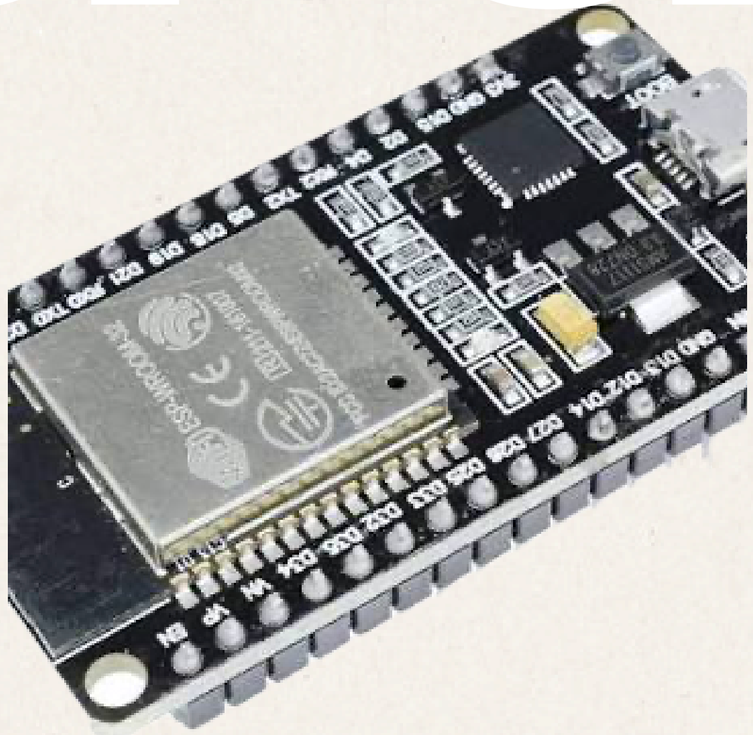
L298N

Speed Sensor

Incremental Optical Encoder

ESP-32

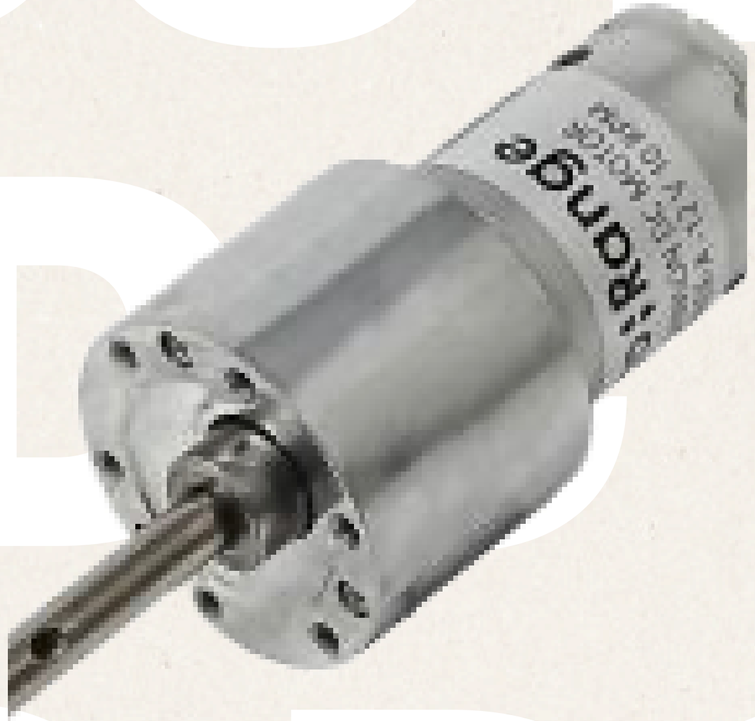
- Used to execute the PID control algorithm and manage the intercepts.
- Uses it’s WiFi compatibility to enable **MQTT communication**, allowing for remote **RPM** control and real-time data publishing.



Specifications: ESP-32 Dev Module

D.C. MOTOR

- Our DC motor is the non-linear element we must control.



Specifications: Johnson (12 V and 300 RPM)

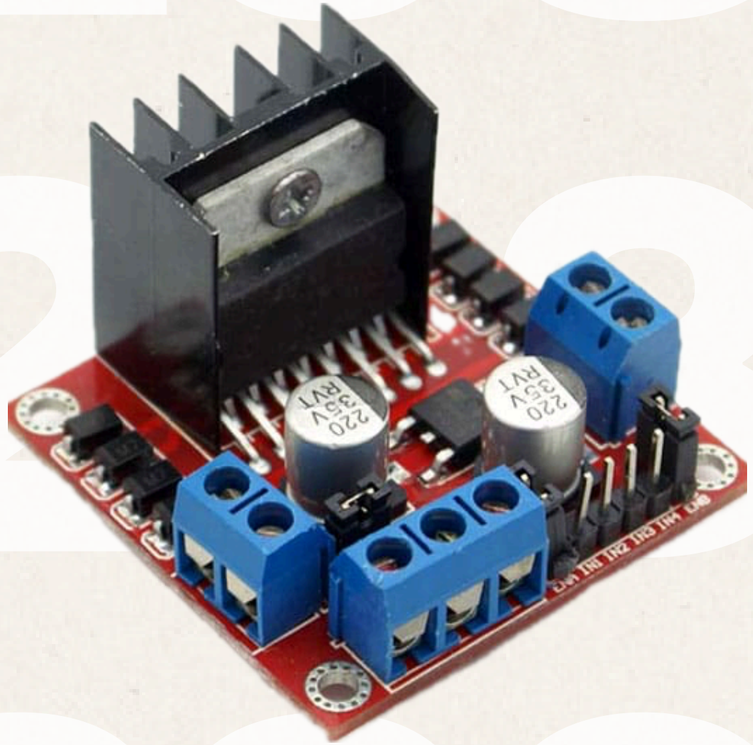
INCREMENTAL ROTARY ENCODER

Measures the motor's actual rotational motion by generating 600 PPR (Pulses Per Revolution). This high-resolution count is used to accurately calculate the Actual RPM via interrupts, forming the vital feedback signal for the PID loop.

Specification: Incremental Rotary Encoder (600 PPR and 4-phase)

MOTOR DRIVER

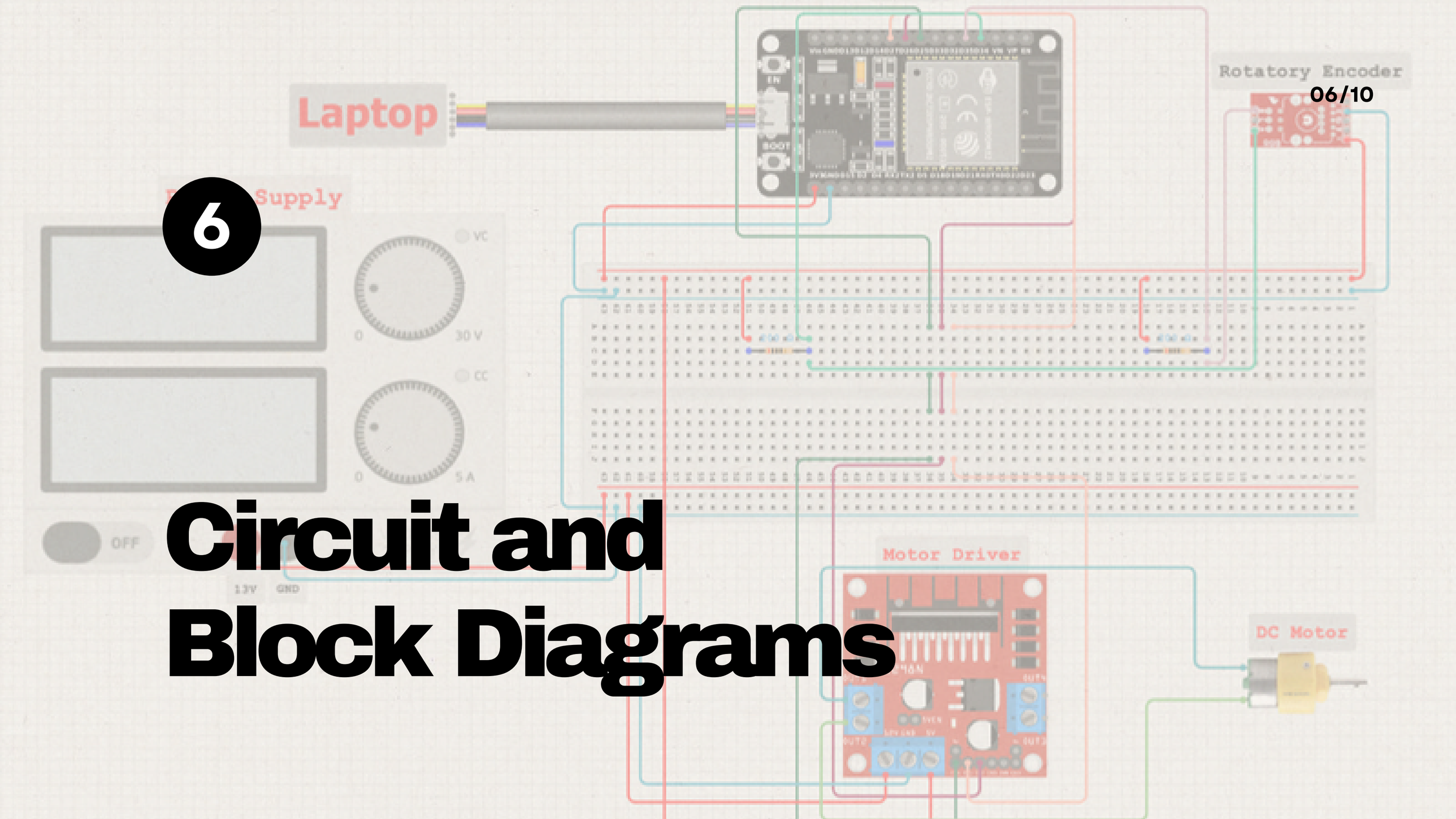
- Acts as an amplifier and interface between the low-power ESP-32 output (PWM and Direction signals) and the high-power requirements of the 12V DC motor.
- It enables the bi-directional control by switching current direction.
- Consumes 1.5V ~ 2V on it’s own.

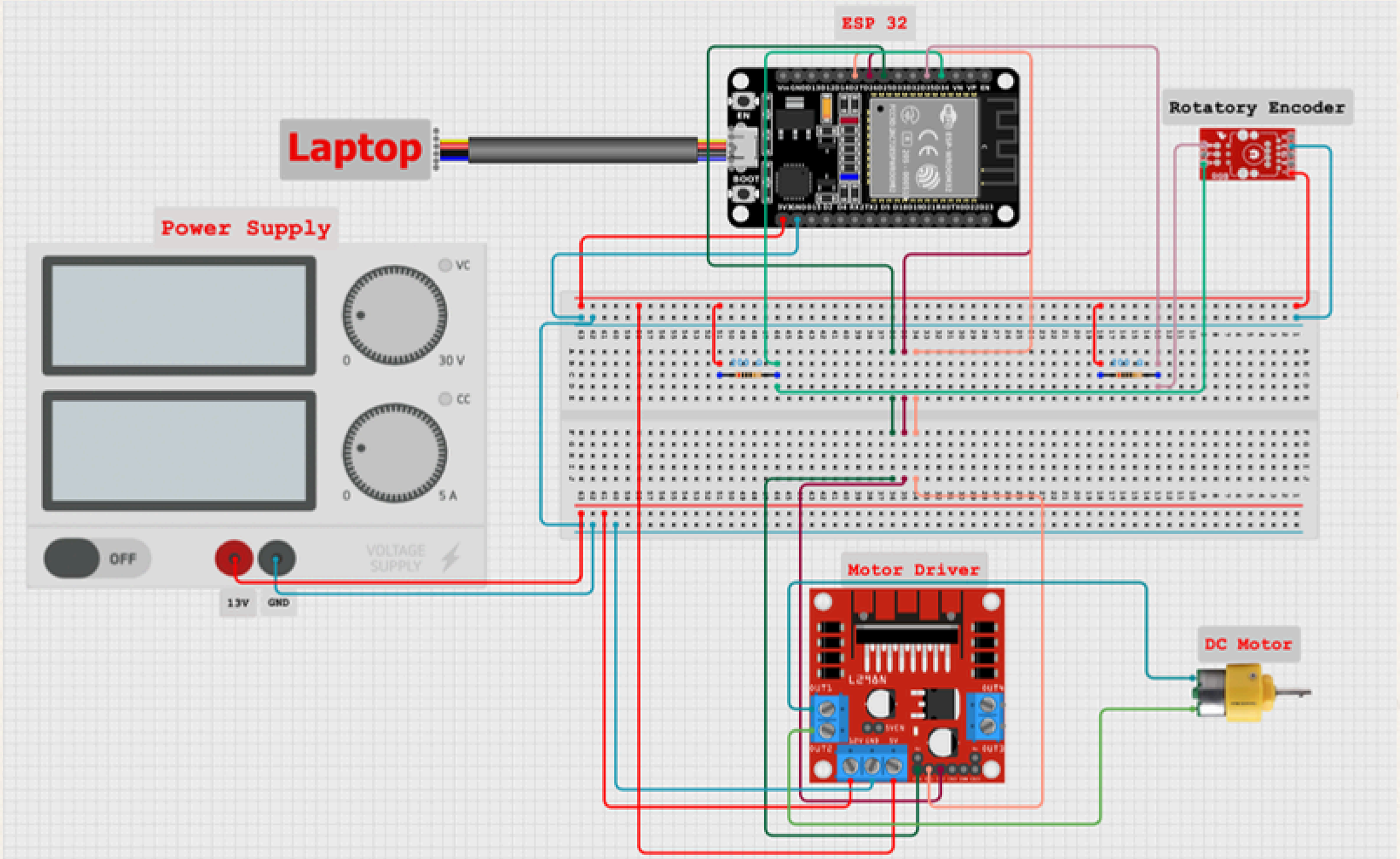


Specification: L298N

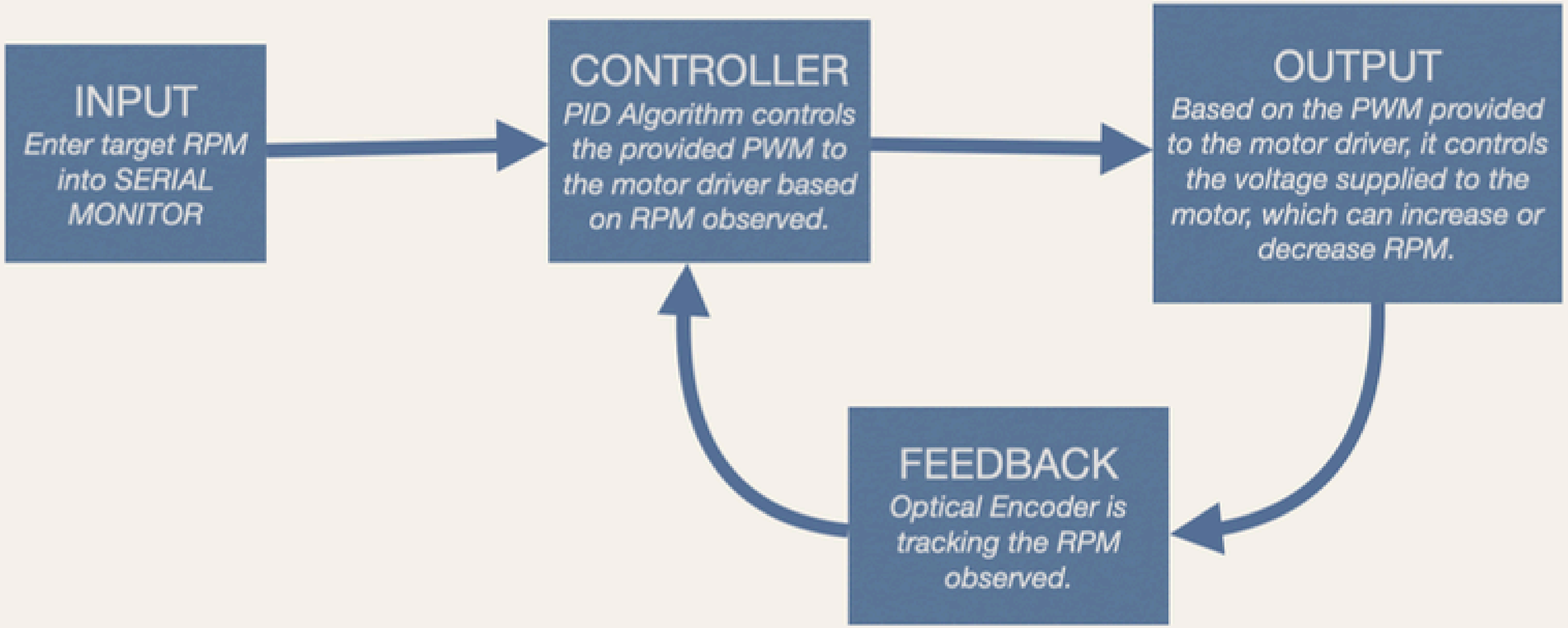
6

Circuit and Block Diagrams




ESP 32
Laptop
Rotatory Encoder
Power Supply
VC
0
30 V
CC
0
5 A
OFF
VOLTAGE SUPPLY
13V
GND
Motor Driver
L298N
DC Motor

BLOCK DIAGRAM



BLOCK DIAGRAM

CALIBRATION & AUTO-TUNING

The calibration and auto-tuning for the PID control of the DC motor.

Calibration & Auto-Tuning

- DC motors are non-linear. This means that voltage and RPM increase at a different rate. A pure PID control would be too slow to control this.
- So to solve this, we basically stored the data for 29 different RPMs all across our testing range.
- Now, we basically adjusting our rates using these points such that we have a base PWM for any given RPM in the given testing range.

Calibration & Auto-Tuning

- We use motor calibration data to predict the base PWM required for any Target RPM.
- We use the measured PWM and RPM table (29 measured points) to model the motor. Then we check where the point lies in the set of all {PWM, RPM} pairs.
- Then we take a linear interpolation between $\{PWM_i\}$ and $\{PWM_{i+1}\}$ to figure out $PWM_{initial}$.
- If the Target RPM falls between two calibration points, the function calculates the exact PWM needed.
- This provides 80-90% of the required power instantaneously. Thus, the PID controller only needs to focus on correcting the small remaining Error caused by load changes or voltage fluctuations, leading to faster response and reduced overshoot.

Calibration & Auto-Tuning

- Auto-Tuning: A Zeigler-Nichols is implemented to automatically calculate the optimal K_p, K_i, and K_d gains, ensuring peak performance and stability without manual tuning.
- Bi-directional Control: Logic is implemented to run the motor in the forward direction (RPM≥0) or the opposite direction (RPM<0).

Calibration & Auto-Tuning

Table 2. Ziegler Nichols Tuning Rules

	K_c	τ_{Int}	τ_{Der}
P	$\frac{K_u}{2}$		
PI	$\frac{K_u}{2.2}$	$\frac{P_u}{1.2}$	
PID	$\frac{K_u}{1.7}$	$\frac{P_u}{2}$	$\frac{P_u}{8}$

Calibration & Auto-Tuning

- PID Equation:

$$Correction = K_p \cdot Error + K_i \cdot \int Error \cdot dt + K_d \cdot \frac{d(Error)}{dt}$$

where,
we scale the error using K_p : ***Proportionality law, P(t)***
we scale the error over time using K_i : ***Integral law, I(t)***
we scale the rate of change of the RPM in wrong direction using K_d : ***Differential law, I(t)***

- So basically, we use this correction to figure out how much to adjust the new RPM based on the recorded RPM.
- $PWM_{final} = PWM_{intial} \pm Correction$

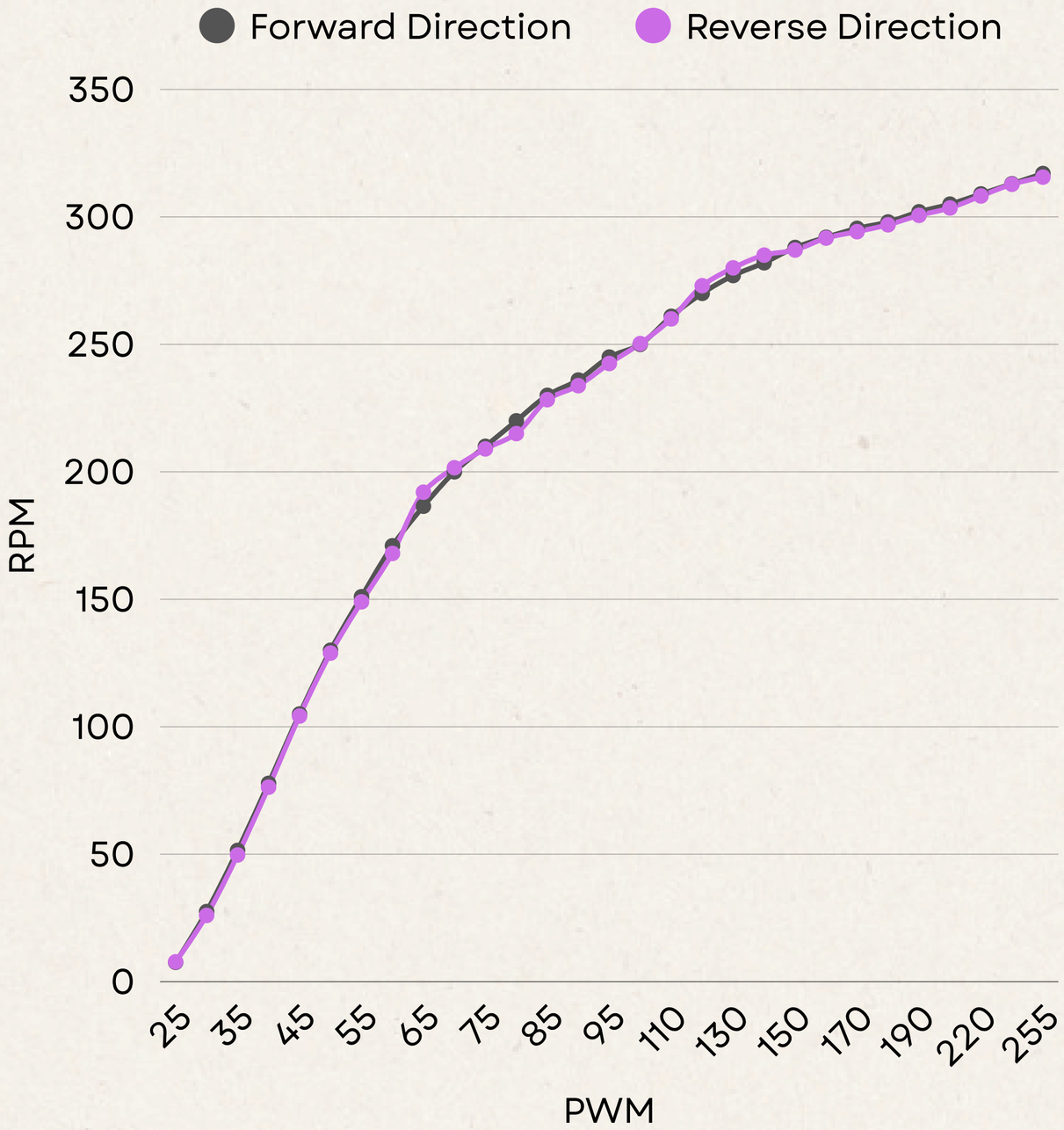
Data: RPM v/s PWM

You can see the data we got from comparing the RPM and PWM values.

The measured PWM vs. RPM relationship is stored in an expanded 29-point calibration table that models the motor's non-linear behavior

A function uses this table to predict and provide a base PWM ($PWM_{initial}$) through linear interpolation for any Target RPM.

This provides 80-90% of the required power instantaneously, allowing the PID to focus on small error corrections.



POSITIONAL CONTROL

The logic for implementation of positional control of DC motor.

CALIB
ATION
AUTO-
TUNIN

POSITIONAL CONTROL

- The rotary encoder gives feedback as pulses per revolution (PPR).
  - PPR (Pulses Per Revolution) :
    - This is the number of high pulses generated per revolution.
    - PPR = 600
  - Counts Per Revolution (CPR) :
    - This is the actual number of counts the ESP32 receives per full turn
    - CPR = PPR × 4 = 2400
- Then, we convert counts to revolutions:
  - revolutions = Δcount / CPR, where Δcount is the number of counts measured.

POSITIONAL CONTROL

- To find the degree of turn,
  - Degrees = (Δcount / CPR) * 360.
    - 1 revolution = 360° = 2400 counts
    - 1° ≈ 2400 / 360 = 6.67 counts
- Every encoder pulse directly tells us how much the motor has turned.
  - angle = (totalCount / 2400) * 360
  - positionError = (targetPosition - currentPosition)
- The PID controller takes this error and adjusts the motor’s PWM output to reduce it.
- The motor keeps turning until the error < 2°

OSITIONAL CONTROL POSITIONAL CO

PCB AND COMPACT APPARATUS

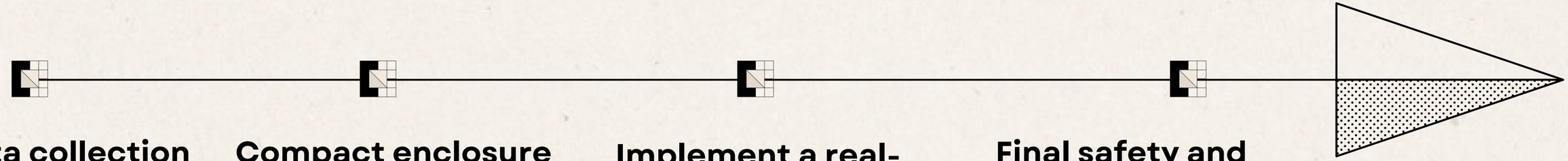
PCB connections and compact enclosure for this project.

PCB AND APPARATUS

- We have successfully implemented the entire control circuit on a custom-designed Printed Circuit Board (PCB).
- This moves us from the breadboard prototype to a finalized, reliable hardware setup.
- Following our plan to finalize deployment, we have started designing the "box" (enclosure) for the system.
- This will house the PCB and all components, creating a robust and complete package for final evaluations.

Future Plan

Here is the plan for the remaining part of our project:



Data collection

Formalize and record the data for the different methods tested for PID-control.

Compact enclosure

Place the designed setup in a compact enclosure.

Implement a real-world use case

Showcase the functionality of our project in a real life use case.

Final safety and robustness testing

Finalize our project for deployment.

Thank you