# PID Control of DC Motor

TEAM 25: SNEHA RAGHAVA RAJU,ABHIJITH ANIL,LAVISHA BHAMBRI,TISHA DUBEY

# Motivation

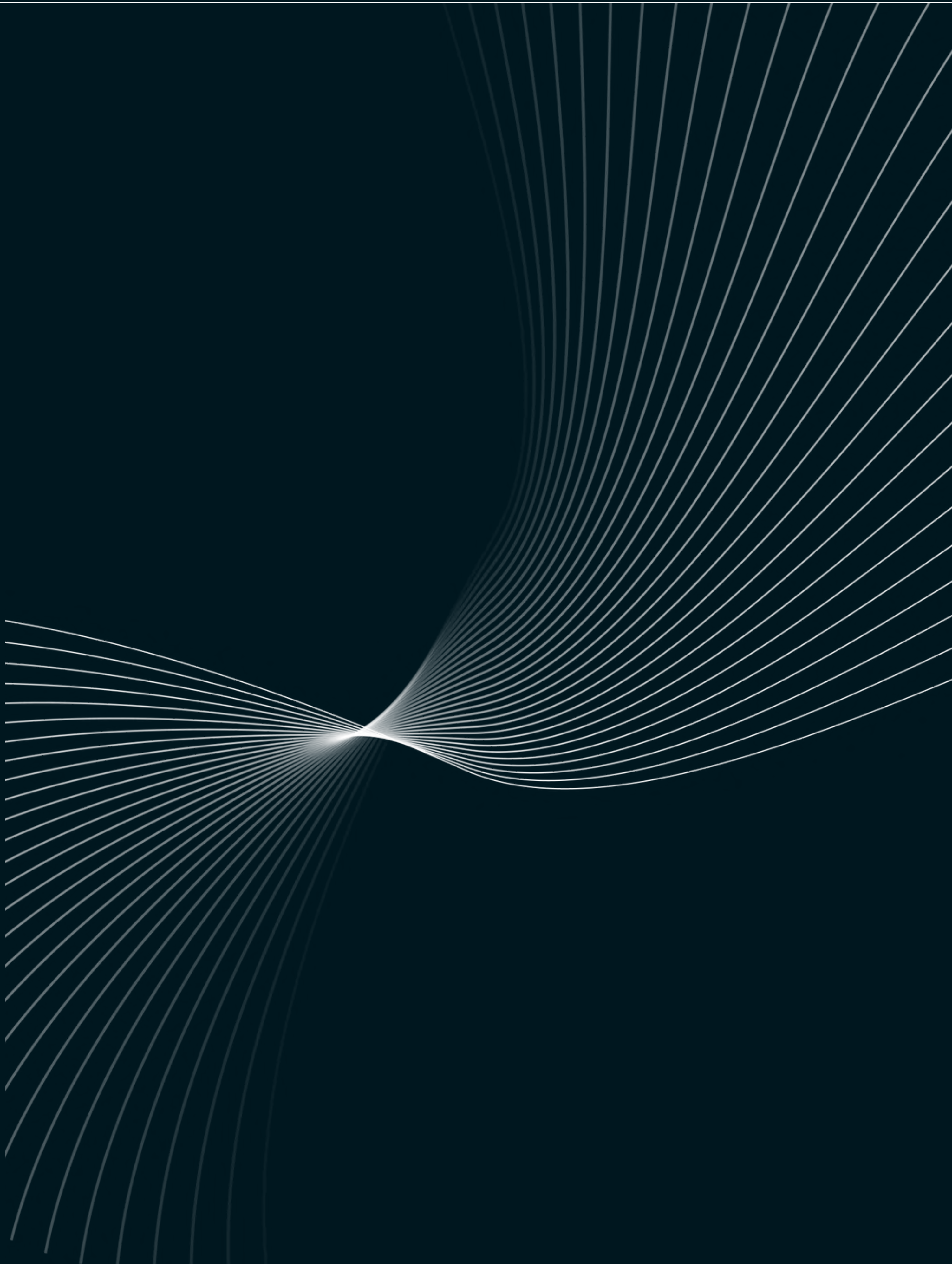
In today's world almost every industrial application uses DC motor. From the paper and textile industry all the way to robotics and cruise control in cars. Control of the position of the angular motor is crucial for the working of many machines in these fields like controlling an arm of a robot.

For example, moving robotic arms to pick stuff in the auto assembly line. Robots arms are now being used in very precise works like performing operations. These require the motor to rotate at a particular angle. Therefore, the control of the position of motors will become more important in the future.

Generally in a DC Motor, speed control can be achieved by varying the terminal voltage but position control of the shaft cannot be achieved efficiently. Hence we will use PID to control the angular position of the DC motor

## Problem Statement

The aim of the project is to reach the target angle using PID logic from any given initial position. We also wish to give the user more room for experimenting with the PID constant along with the option of providing the target angle. Along with these, the project  includes a dashboard to show the collected data, statistics based on the previous results of PID tweaks.



# Project Summary

- The experiment is to create a system consisting of a DC motor whose angular position is controlled by an ESP32 board.
- The system also has an encoder attached to the DC motor that can be used to find the angular position of the main shaft of the DC motor.
- The ESP32 sends the sensor data to the cloud.
- A web application visualizes the passed sensor data, the current sensor output and the live feed of the motor. It also contains an option to give commands to rotate the motor by a certain angle and change PID constants.
- The user can give the command and observe in real time how the result is achieved.

## Methodology

Our system includes:
Hardware: A microcontroller running PID algorithm controlling a motor shaft. Its target angular position can be provided remotely via the dashboard and supporting statistics and analytics on the sensor readings viewable from the dashboard over the internet.

Platforms: Thingspeak is used to communicate from the microcontroller. Database is used to store user and slot data. Cloud platform to host the dashboard.

Dashboard: A dashboard to provide the target angular position and PID constants to conduct a trial and observe the results on a graph and a live stream of the experiment.

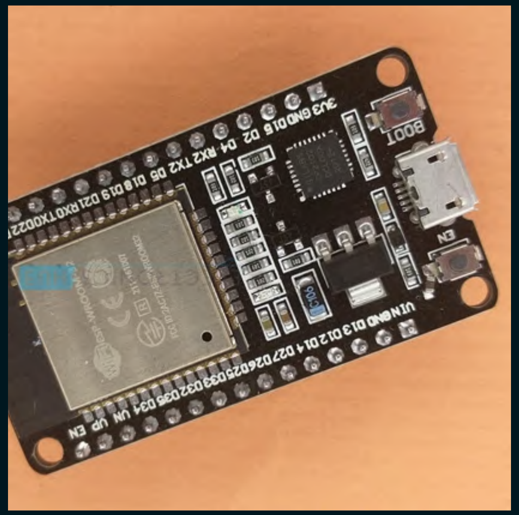
To develop the system within the given timelines, we used the Agile Methodology. The Agile methodology is a way to manage a project by breaking it up into several phases.
We divided the project into 6 sprints:-
Sprint 1 - Choosing the hardware
Sprint 2 - Hardware Integration and PID Logic
Sprint 3 - Thingspeak integration and pushing data to Om2m
Sprint 4 - Dashboard Frontend and Backend
Sprint 5 - Setting up the ESP32 Cam and final caliberation along with data analysis

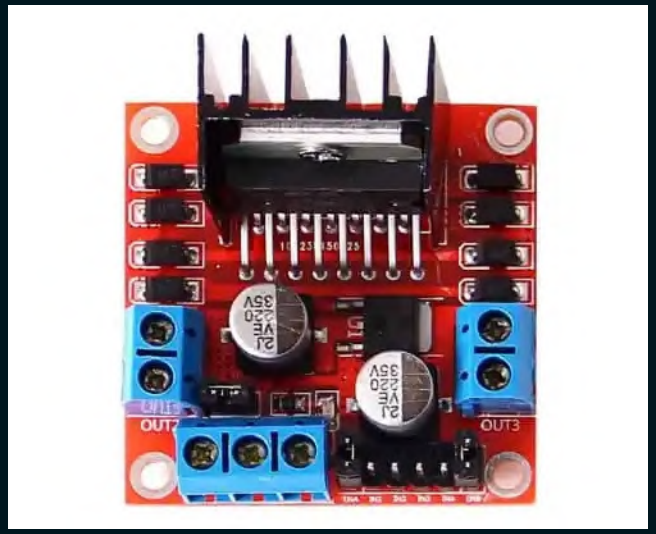
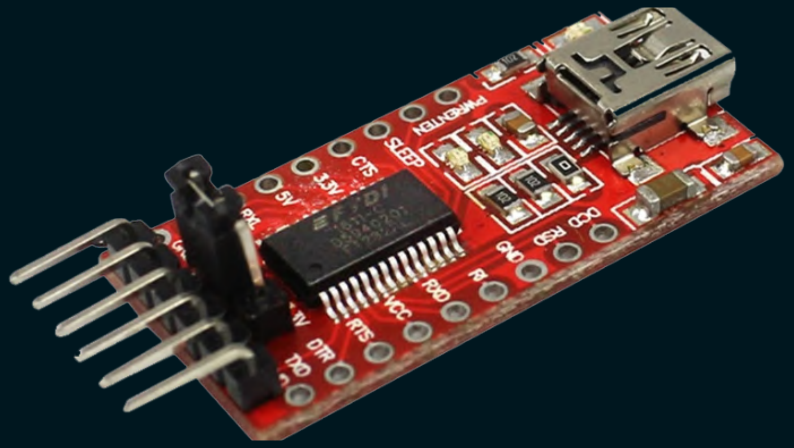
# Sprint #1

CHOOSING THE HARDWARE.

The hardware components which we used are :

- ESP32 Microcontroller
- DC Motor with an encoder
- L298 Motor driver
- DC adaptor for external power supply
- ESP32 Cam
- USB to TTL



Pitch

# Sprint #2

HARDWARE INTEGRATION AND PID LOGIC.

- Soldering of hardware components on Zero PCB.
- Logic or PID.

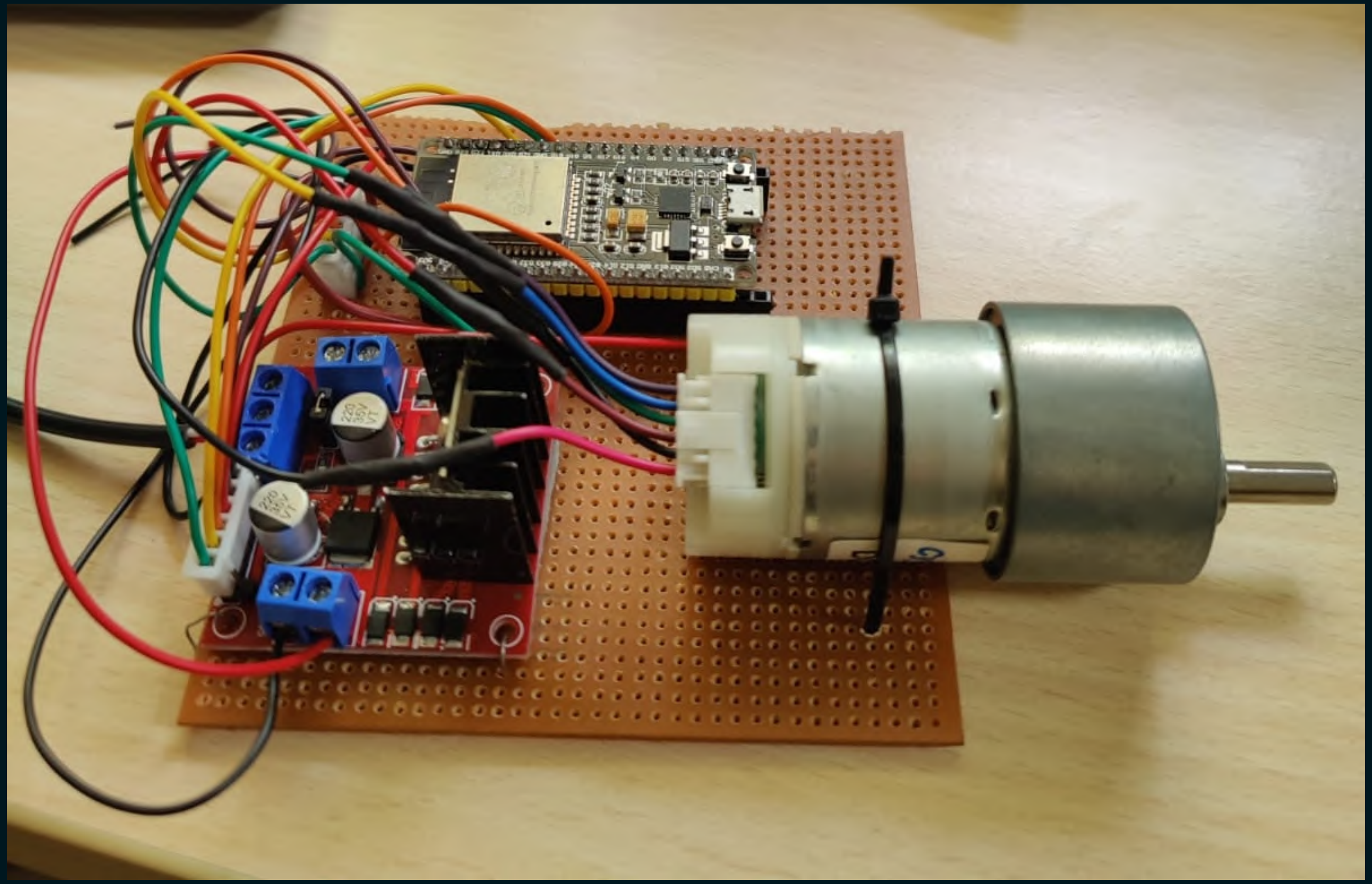
The PID calculations has been done as follows:

Calculation:

```
// error
int e = pos - target;

// derivative
float dedt = (e-eprev)/(deltaT);

// integral
eintegral = eintegral + e*deltaT;
```

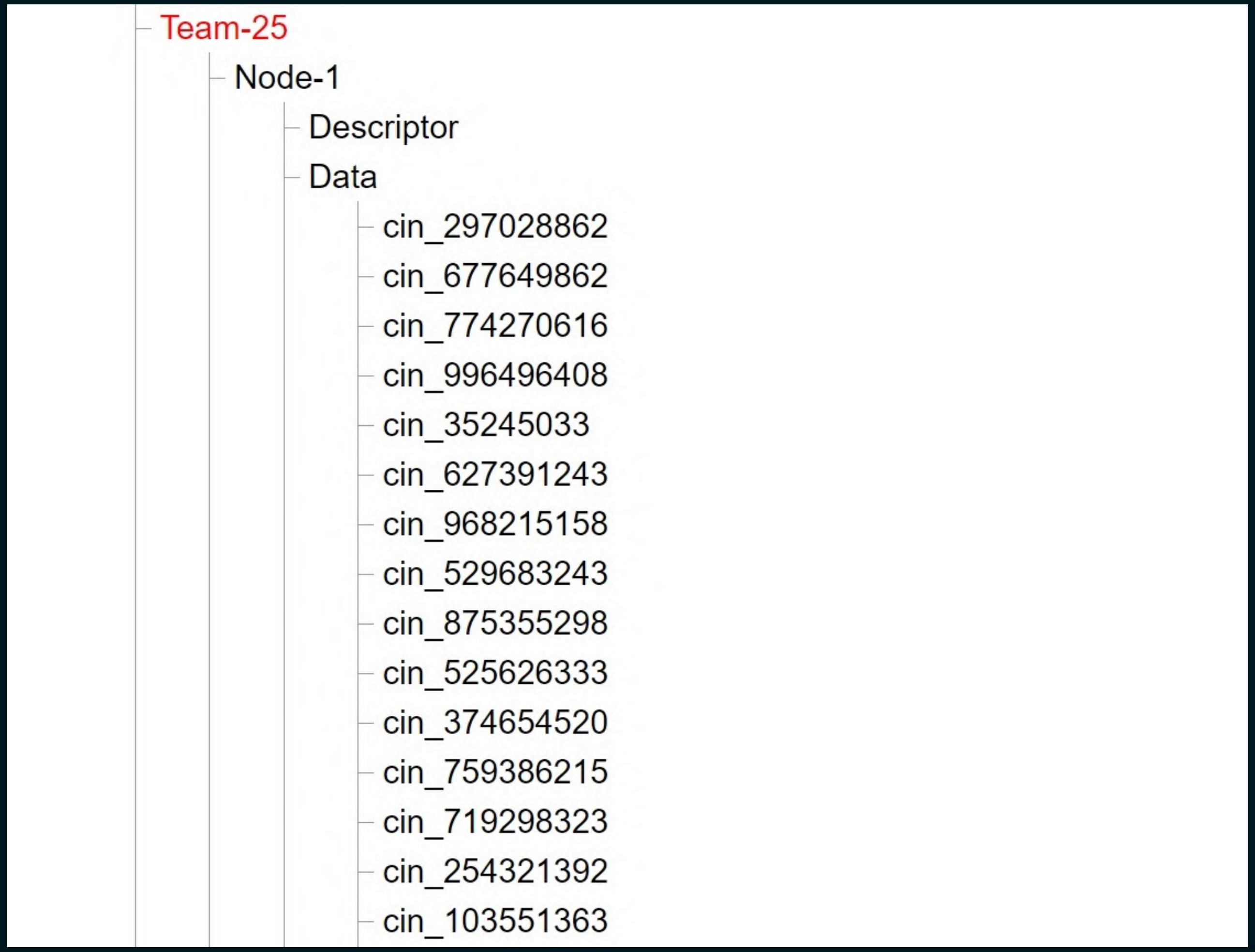


The PID terms are calculated according to the formula

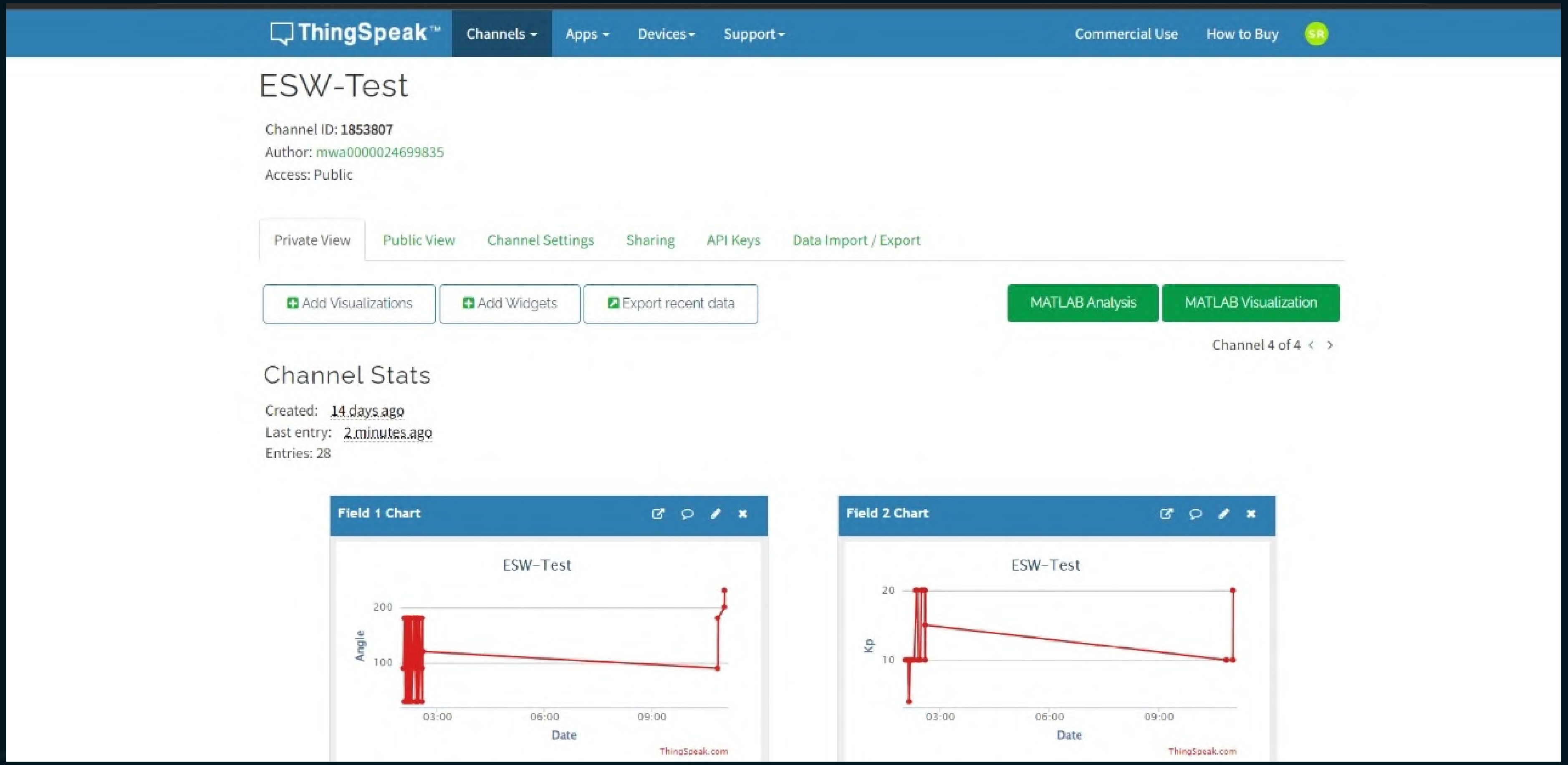
```
// control signal
float u = kp*e + kd*dedt + ki*eintegral;
```

# Sprint #3

THINGSPEAK INTEGRATION AND PUSHING DATA TO OM2M.



- Team-25
  - Node-1
    - Descriptor
    - Data
      - cin_297028862
      - cin_677649862
      - cin_774270616
      - cin_996496408
      - cin_35245033
      - cin_627391243
      - cin_968215158
      - cin_529683243
      - cin_875355298
      - cin_525626333
      - cin_374654520
      - cin_759386215
      - cin_719298323
      - cin_254321392
      - cin_103551363



ESW-Test

Channel ID: 1853807
Author: mwa0000024699835
Access: Public

Channel Stats

Created: 14 days ago
Last entry: 2 minutes ago
Entries: 28

Om2m

Thingspeak Testing test

## Sprint #4

DASHBOARD FRONTEND AND BACKEND.

The dashboard is implemented using MernStack and the deployment is done online using MongoDB Atlas. The backend server has been developed using the ExpressJS library in NodeJS.

**Backend** - The backend for the dashboard stores the user details, and the slot booking details.

**Frontend** - The front end was created to give users live feedback of the experiments, showing the plot of target angle and current position and live twitch stream of DC motor.

The following features have been implemented:

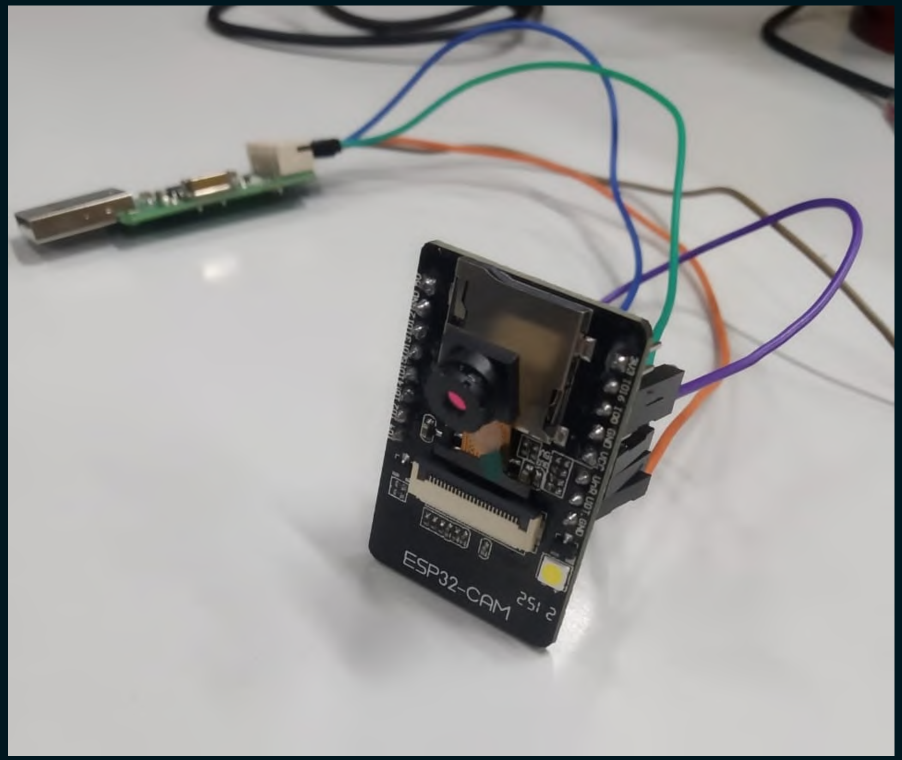
- One login at a time.
- The user can login only in the allotted slot which he/she has booked while registration.
- The user gets logged out as soon as slot time ends.

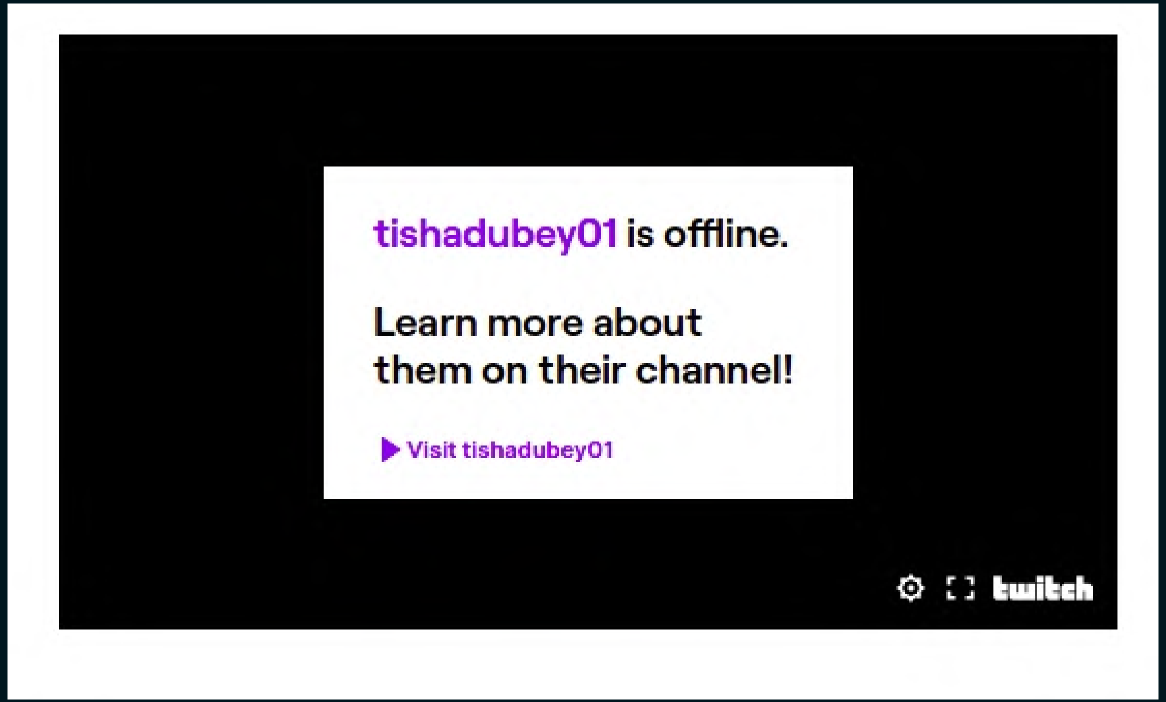
SECURITY AND AUTHORIZATION

- Communication between the frontend and backend is encrypted through HTTPS.
- User login/registration along with JWT was added for authorization. A MongoDB database provider Mongo Atlas was used for storing credentials

# Sprint #5

SETTING UP THE ESP32 CAM AND FINAL CALIBERATION ALONG WITH ANALYSIS OF DATA.





tishadubey01 is offline.

Learn more about them on their channel!

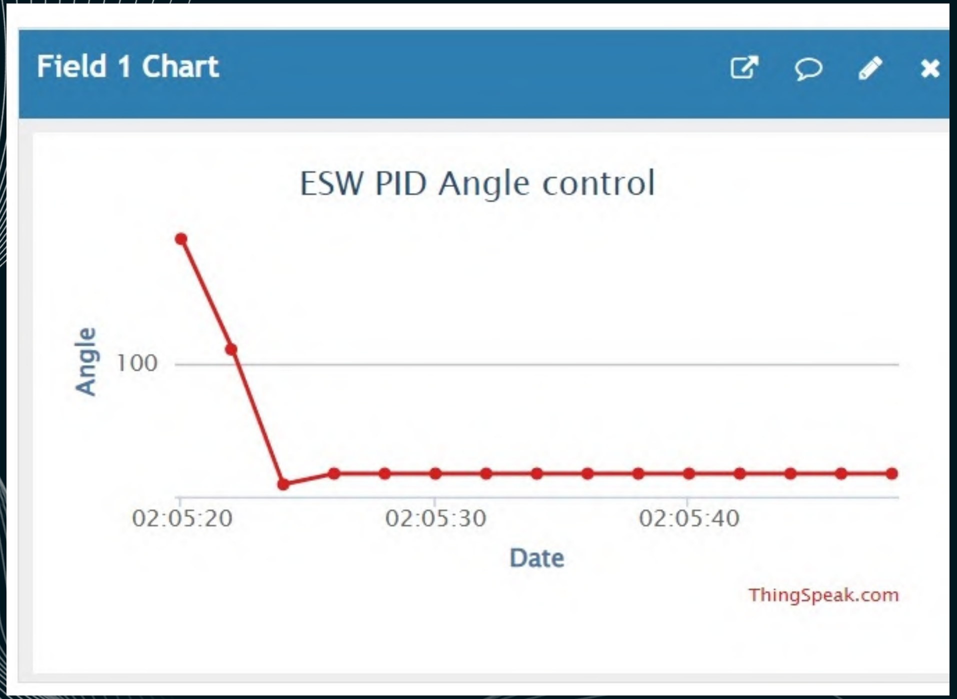
▶ Visit tishadubey01

The USB is connected to the PC for the power supply. During live streaming the camera lags a bit due to low power supply.

Integrating the ESP32 CAm with the OBS and twitch TV for live streaming

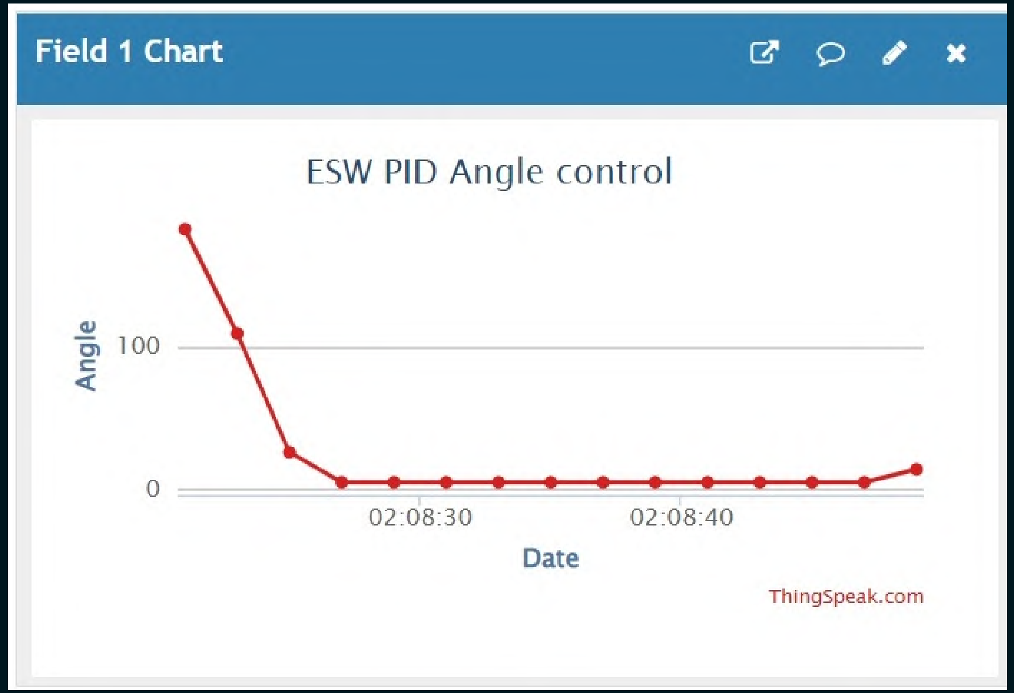
# Caliberation of data

The data has been caliberated manually using the protractor and also by comparing it with the original angle by providing different values of constants.
The motor shaft is attached to a custom-made protractor with a pointer which allows the users to visually verify the angular position of the shaft.



Kp: 10, Ki: 0.4, Kd: 0.125

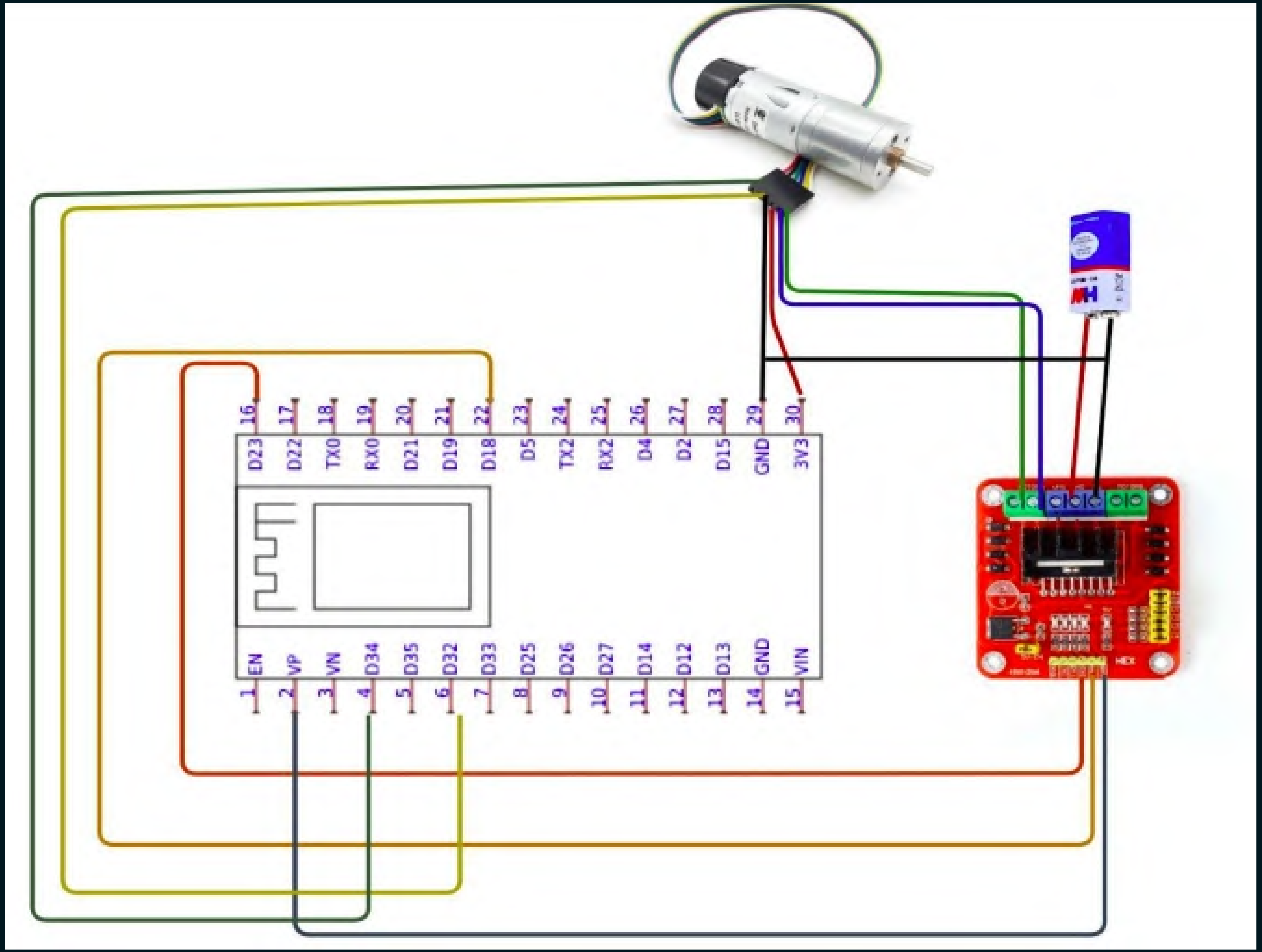
target: 30, reached: 30



Kp: 4, ki: 0.4, kd: 0.125

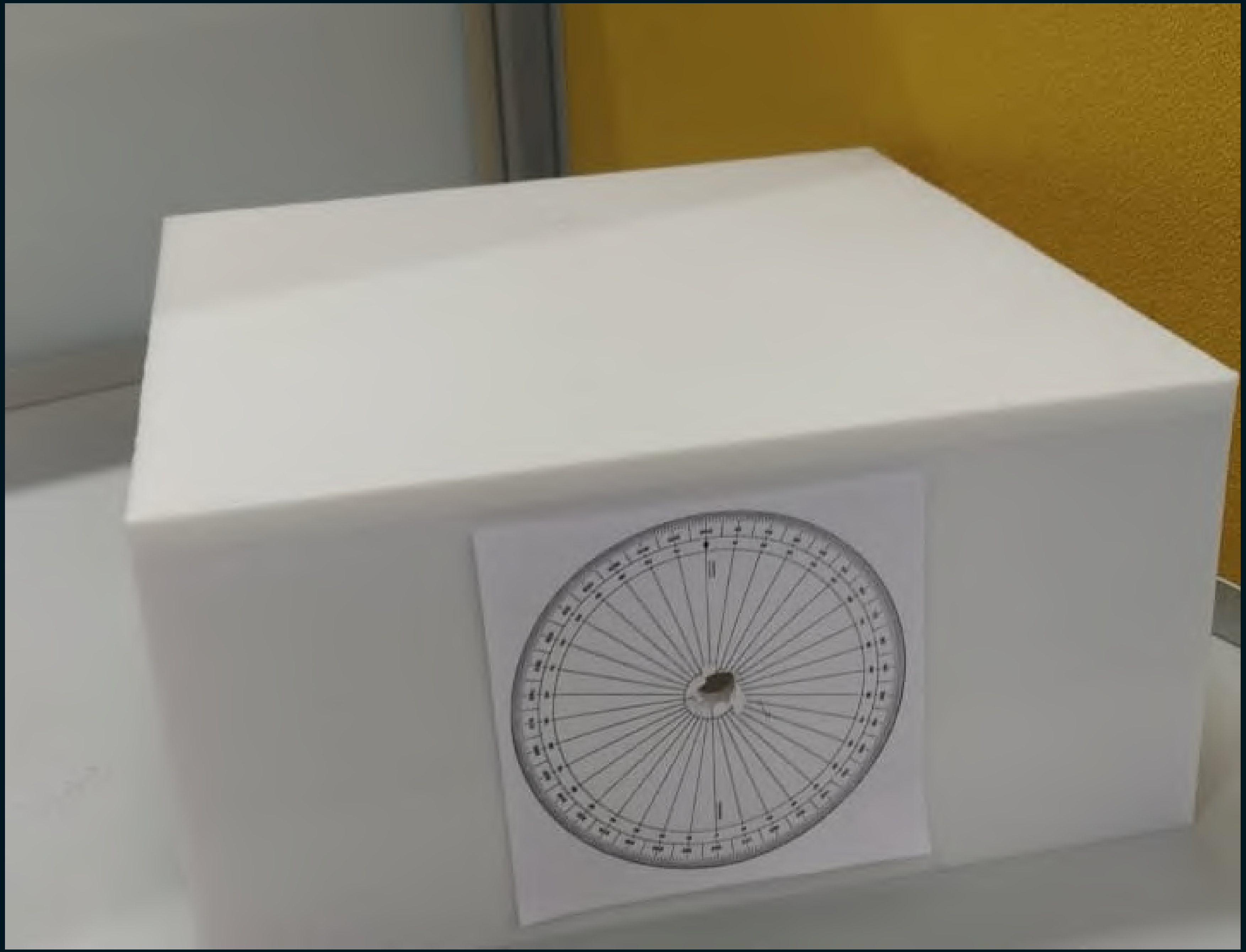
target:30, reached: 14

# Circuit Diagram



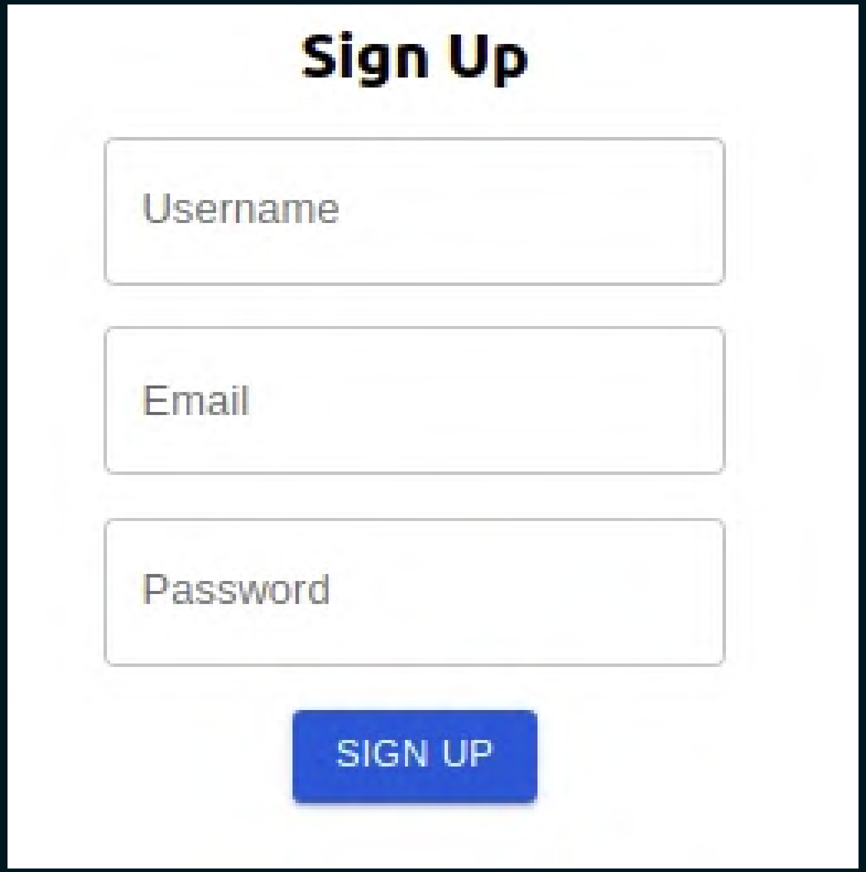
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
D23 D22 TX0 RX0 D21 D19 D18 D5 TX2 RX2 D4 D2 D15 GND 3V3
EN VP VN D34 D35 D32 D33 D25 D26 D27 D14 D12 D13 GND VIN
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

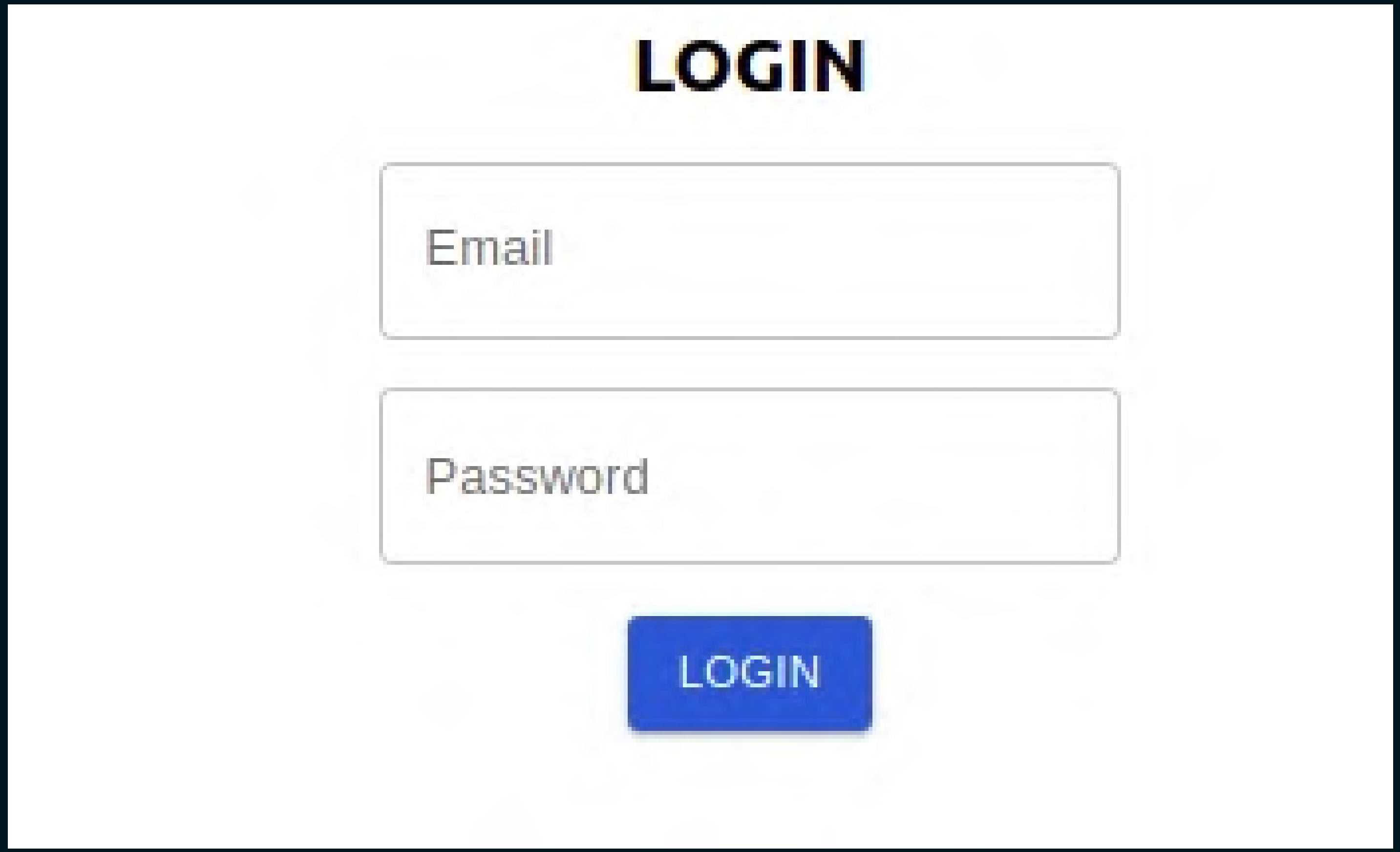
## Enclosure



Pitch

# Dashboard

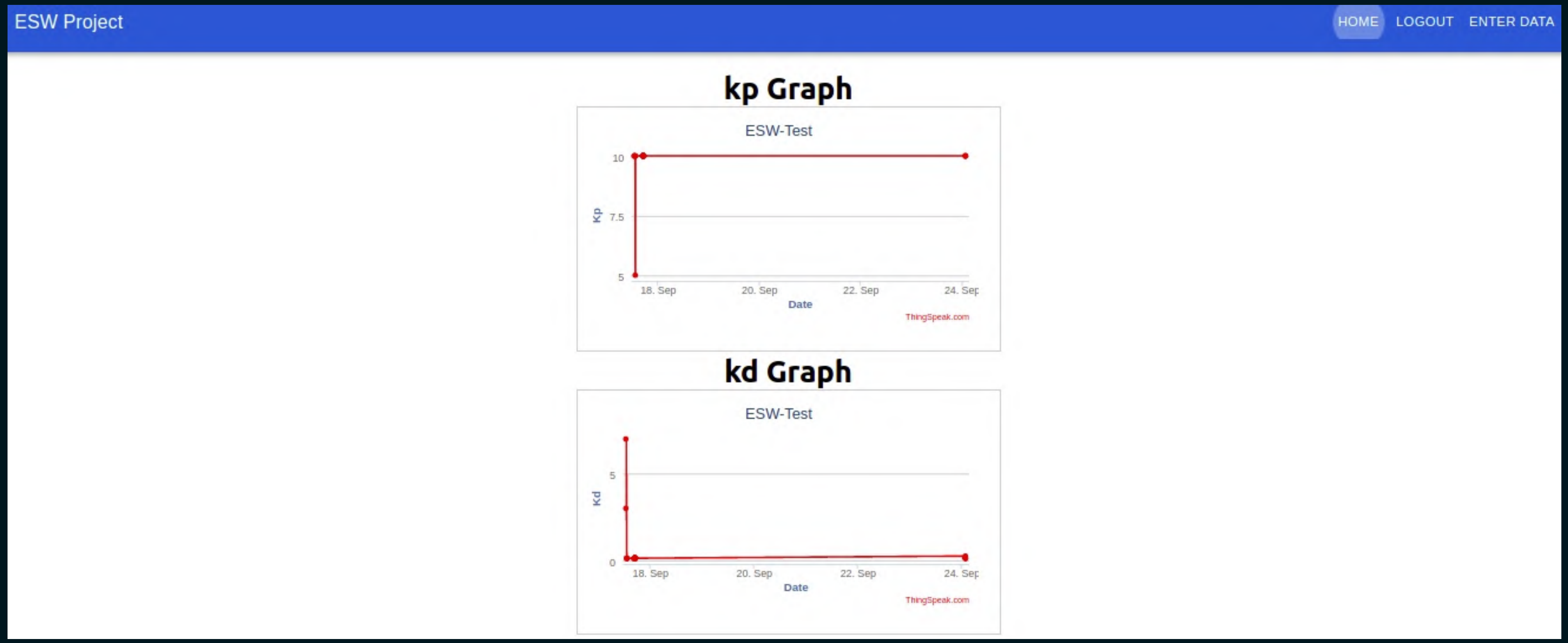




Pitch

# (Contd.)

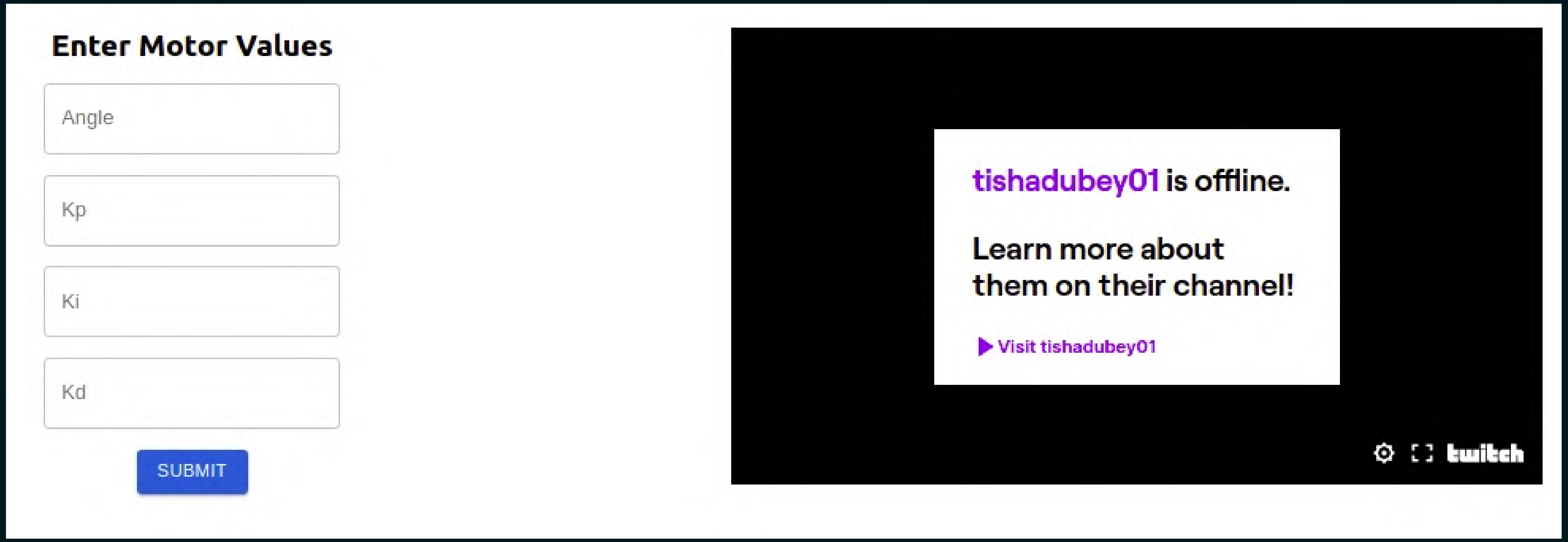
Graphical analysis on the dashboard. The graphs has been taken directly from the Thingspeak.


ESW Project

HOME LOGOUT ENTER DATA

kp Graph

kd Graph


Enter Motor Values

Angle

Kp

Ki

Kd

SUBMIT

tishadubey01 is offline.

Learn more about them on their channel!

Visit tishadubey01

Input angles and the livestream from twitch.tv

## Conclusion

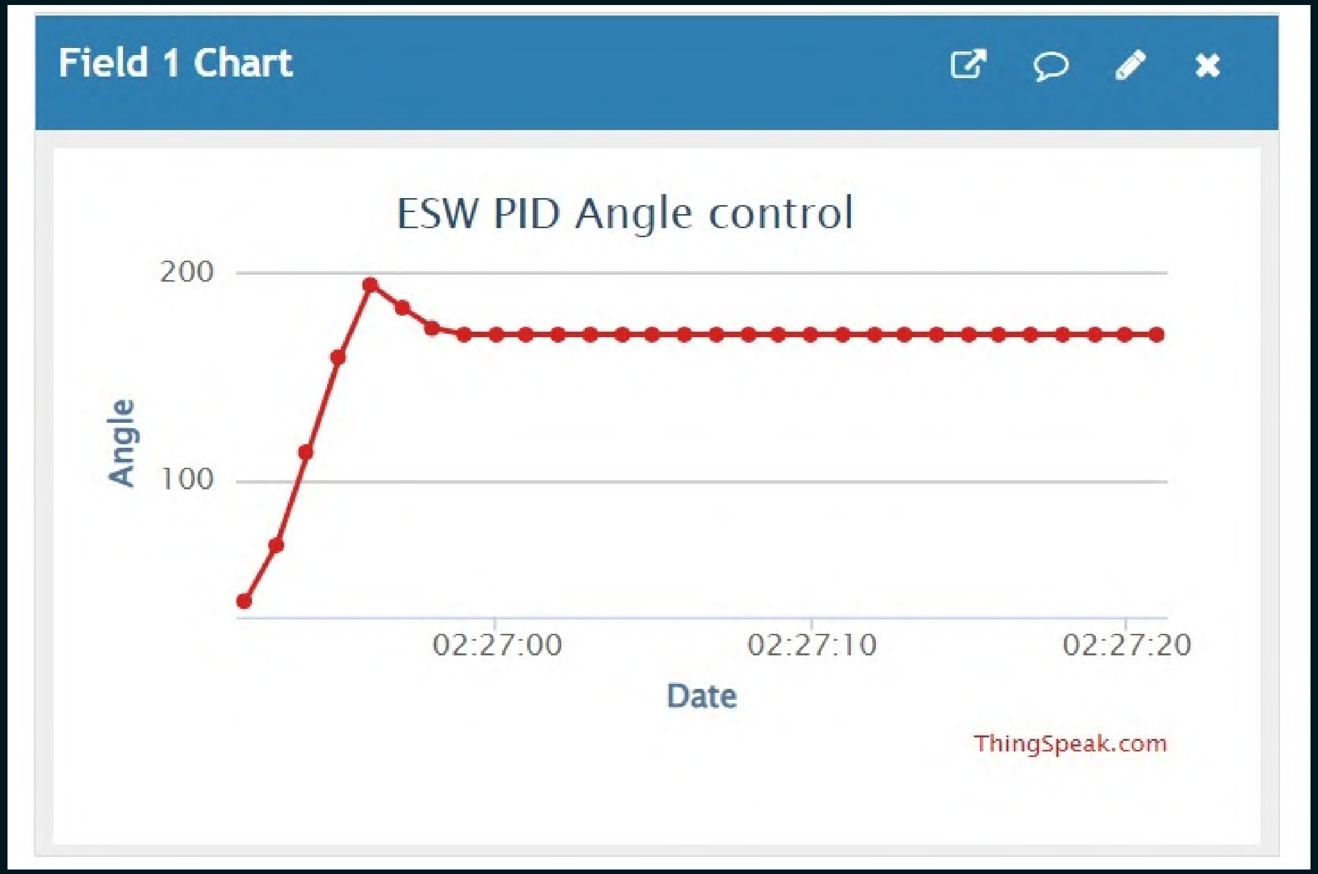
The average convergence time was less than 1 second approximately.
The final error was in the range ±2 degrees i.e. ±0.55%.

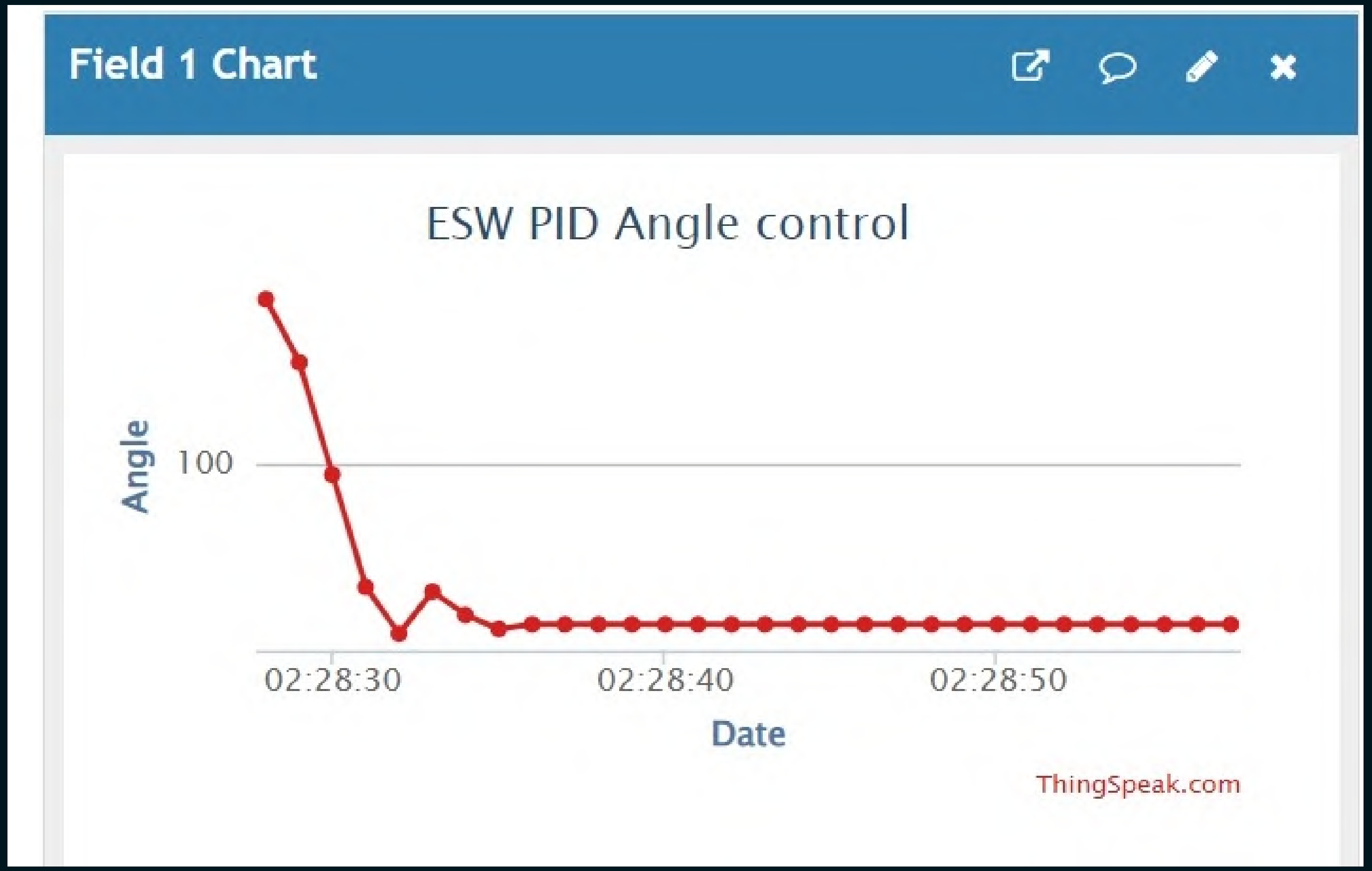
The PID algorithm works by comparing the current angle to the target angle and generating the error and using the PID values to correct it. The PID constants dictate how much each part of the PID sum is represented. The Proportional constant is the most important one and the Derivative and Integral constants act as further fine tuning.

Arriving at the best PID constants was done through trial and error as shown below:

## Demo:

Application Link - https://esw-team-25.herokuapp.com/home
GitHub Link - https://github.com/Tishadubey01/esw-pid-controller



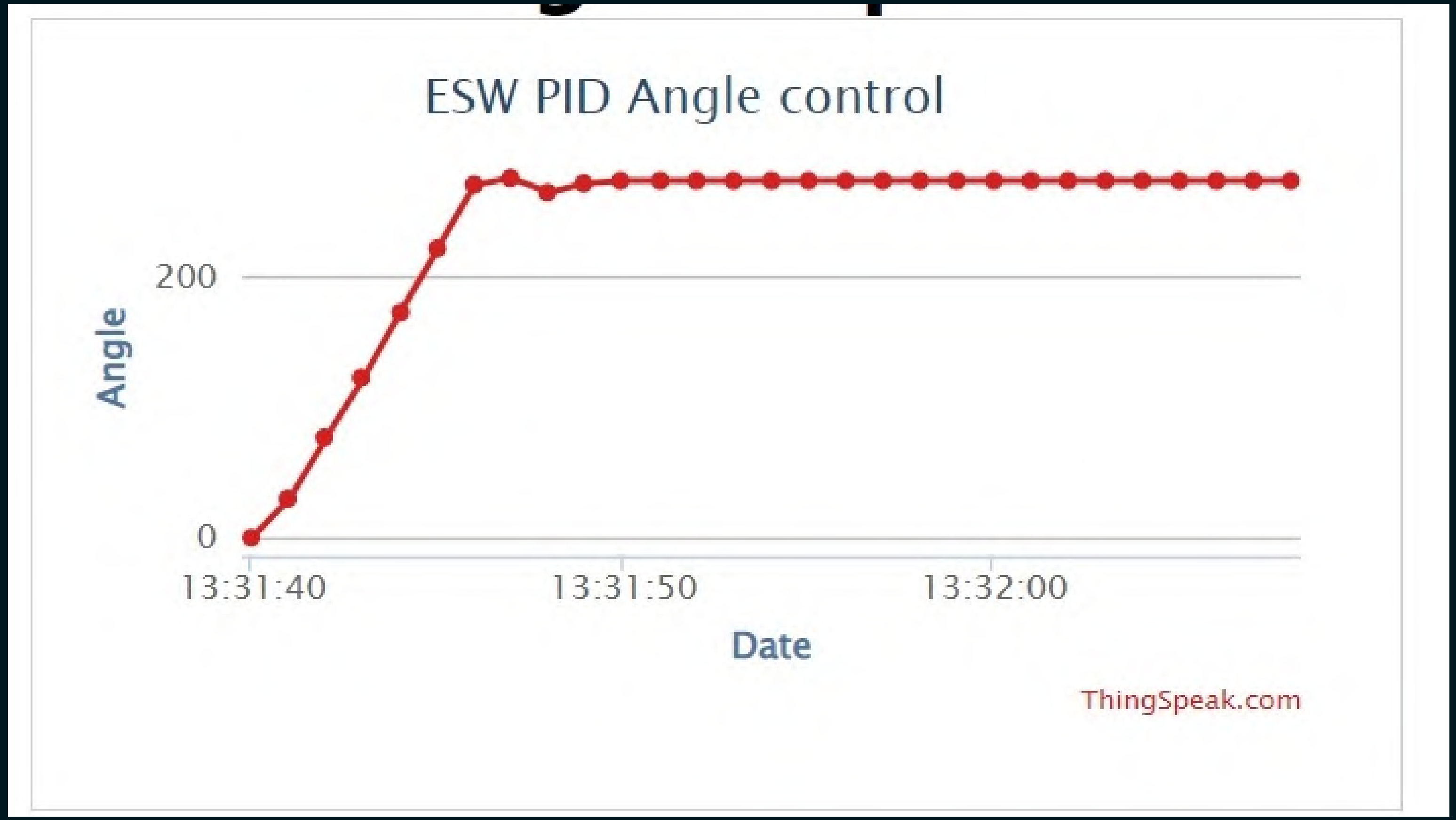


Target: 180, Final Angle: 170 so Error: 10

With the constants as -> kp:10, ki:0.4, kd:0 , there is some overshooting of the angle.

Target: 30, Reached Angle: 30, kp:20, ki:0.5, kd:0

In this case there is some damping in the angle values.



Target: 270, Reached Angle: 274, Error: 4

For kp:20, ki: 0.4, kd: 0.125

For these constants the overshooting and damping are minimized and the error is small, so they are taken as the ideal constant values.

## Contributions

Contribution of each member

Thingspeak Integration

Building the circuit

Hardware code

Building the circuit

Dashboard Frontend

Hosting the website

Dashboard Backend

Camera Integration

Zero PCB

# Thank you.