Ain Shams University

Faculty of Engineering

Mechatronics Engineering Department

# Automatic Control (MCT211)

## Electrical oven project

| Name | ID |
|---|---|
| Alaa Elsayed Metwally Mohamed | 1809361 |
| Ahmed Yasser Ahmed Abdelhamid | 1804679 |
| Ali Abdelhakem Mahmod Hussin | 1805347 |
| Mohamed Emad Mohamed Hussien | 1802245 |
| Mohamed Emad Mohamed Moslem | 1806770 |
| Mohamed Abdulaziz Farrag Ahmed | 1805858 |

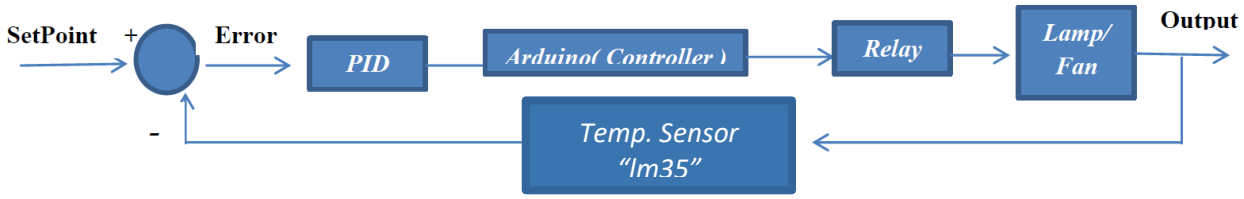
## List of contents:

## 1. Objective and Way of working:

The main target of this project is to design a control system that can keep the temperature of the system at any desired set point using a closed loop system where:

- ❖ The desired temperature represents the set point.
- ❖ The error in the system is represented by the difference between the desired temperature and the feedback signal (actual temperature measured by the temperature sensor ).
- ❖ we use the Arduino as the controller in the system.
- ❖ The 2-channel relay is used in the transition between the two states of heating (lamp on – fan off) and cooling (fan on – lamp off).

## 2. The Block Diagram




## 3. Control Action and final control element

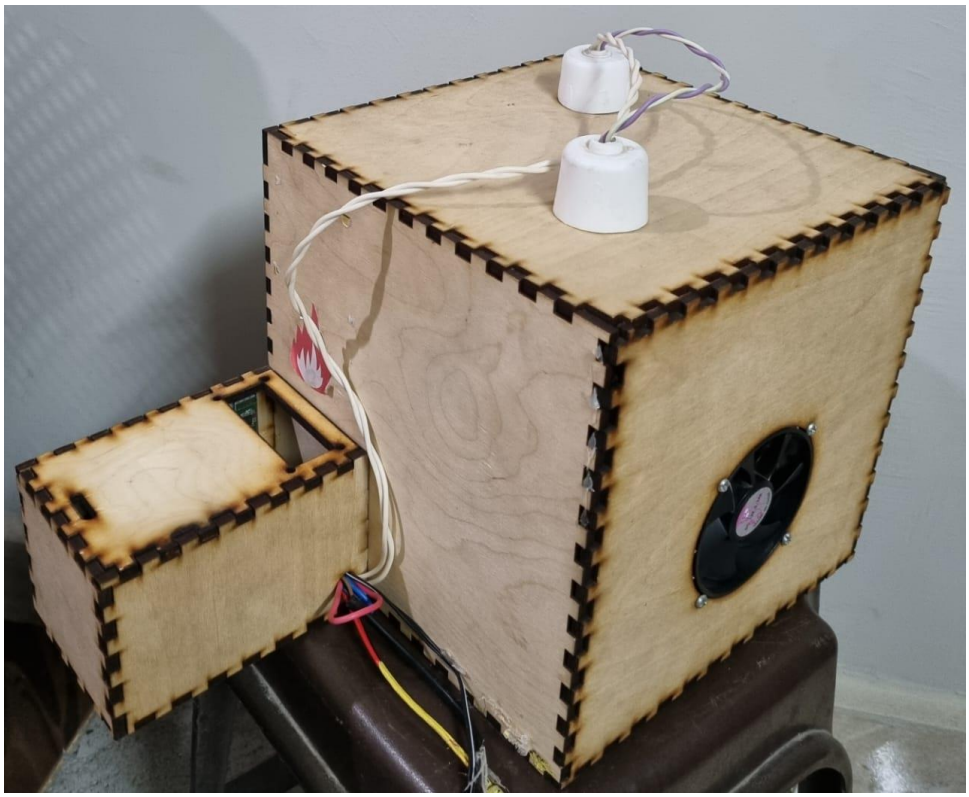
The Arduino(controller) compares the output value (temperature) with the reference input by user, so if the temperature is more than the desired value the controller makes the the fan on and the heating lamp off and reverse this action when the temperature is below the desired value and the set point.

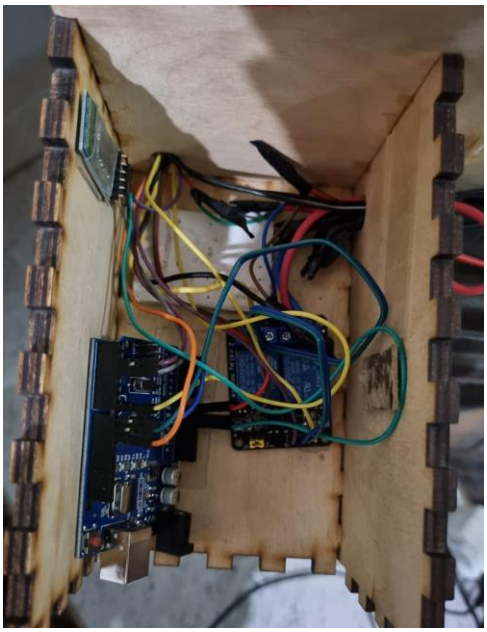
- ❖ Final control element (Magnetic relay ,Fan & lamp): take the signal from Arduino to switch on and off according to the set point

- ❖ Sensor (Temp. sensor "lm35") : detects the temperature in the box and give this signal back to the controller to take action and send signal to the controlled element

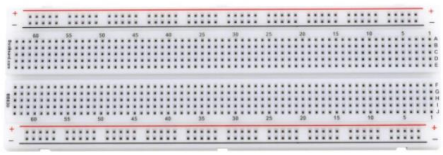
- ❖ Controlled variable (Temperature): controlled by the fan (cooling) and Lamp (heating)

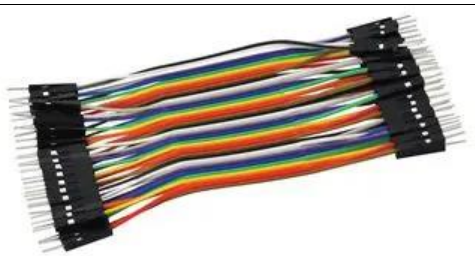
# 4. Overall real hardware system picture





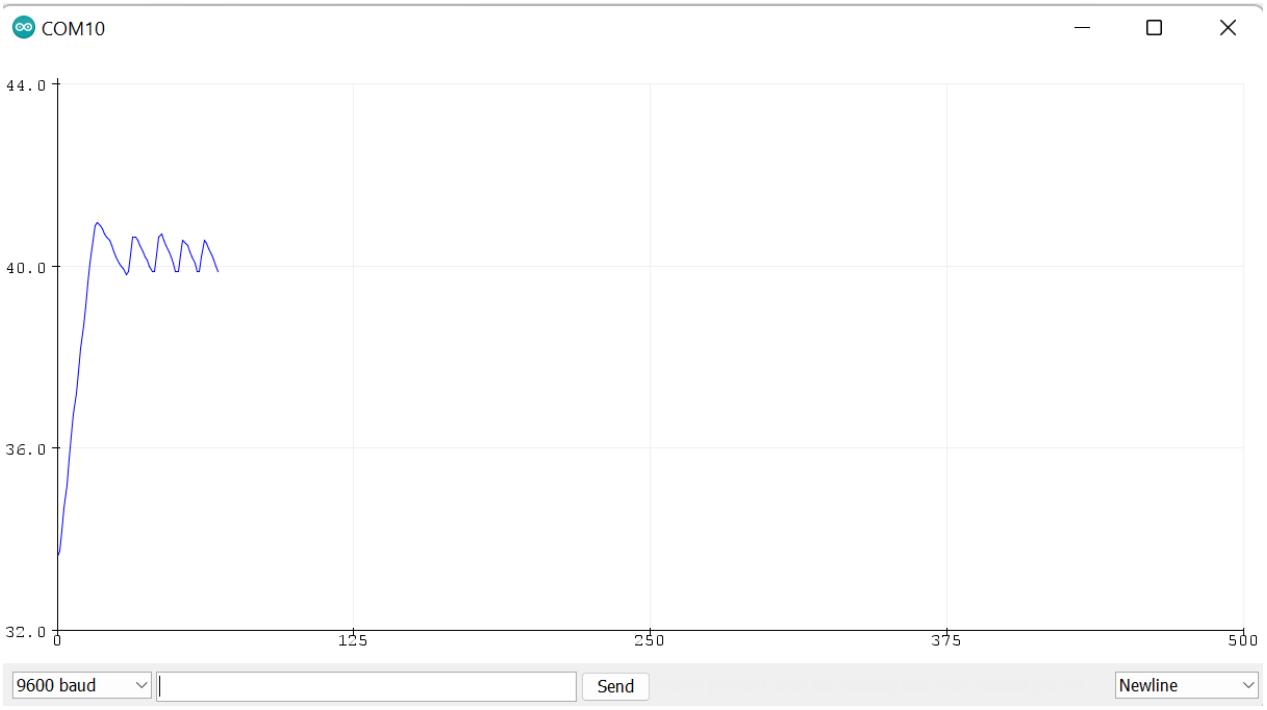
## 5. Components

| Name & Its Function | Picture |
|---|---|
| **1 .Arduino uno :** is able to read inputs and turn it into an output |  |
| **2 Dual-Channel Relay :** are electrically isolated from the controlling input. The relays can be used to switch higher voltage and current loads than a microcontroller can traditionally accomplish. |  |
| **3. Breadboard :** allows for easy and quick creation of temporary electronic circuits or to carry out experiments with circuit design. |  |
| **4. Temperature Sensor:** senses the temperature |  |
| **5. Lamp :** used as source of heat |  |

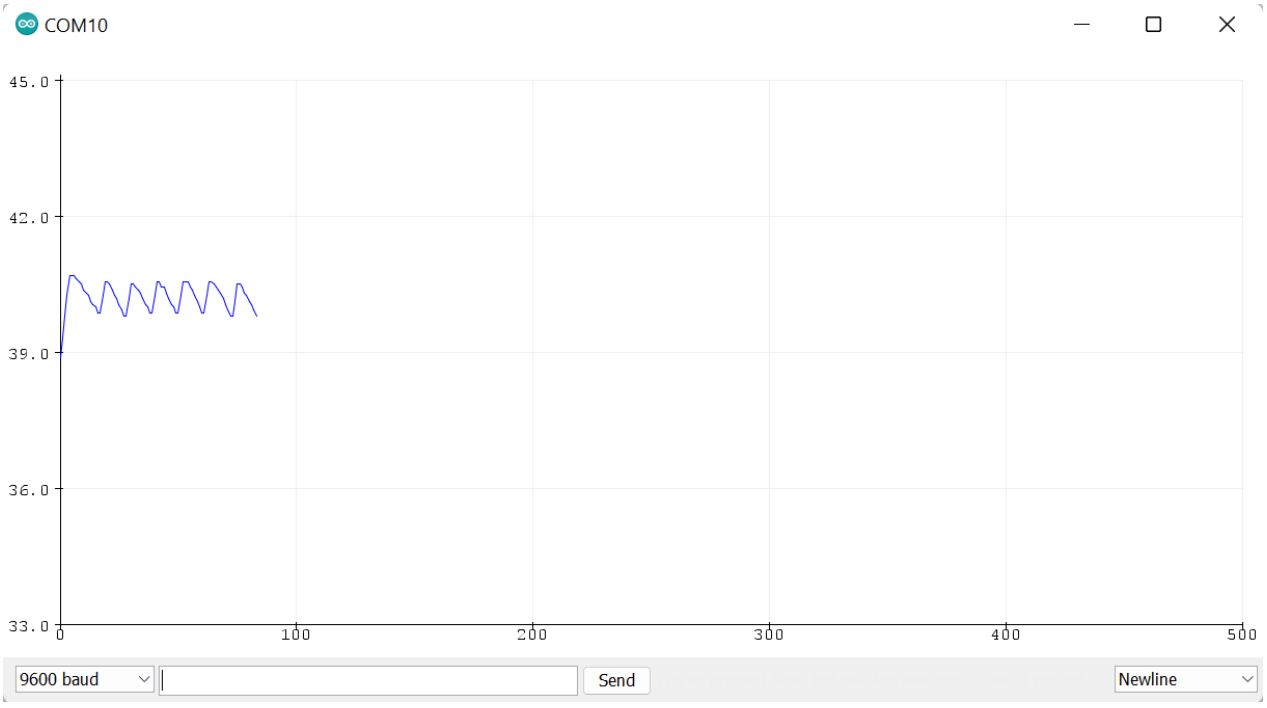
| | |
|---|---|
| **6. Fan :** used to decrease the temperature of the system |  |
| **7. Electric Wire for Lamp :** used to connect the lamp with the source of the voltage |  |
| **8. Wires :** used to create the circuit of the system and connect the components with each other |  |

# 6.Trial Graphs

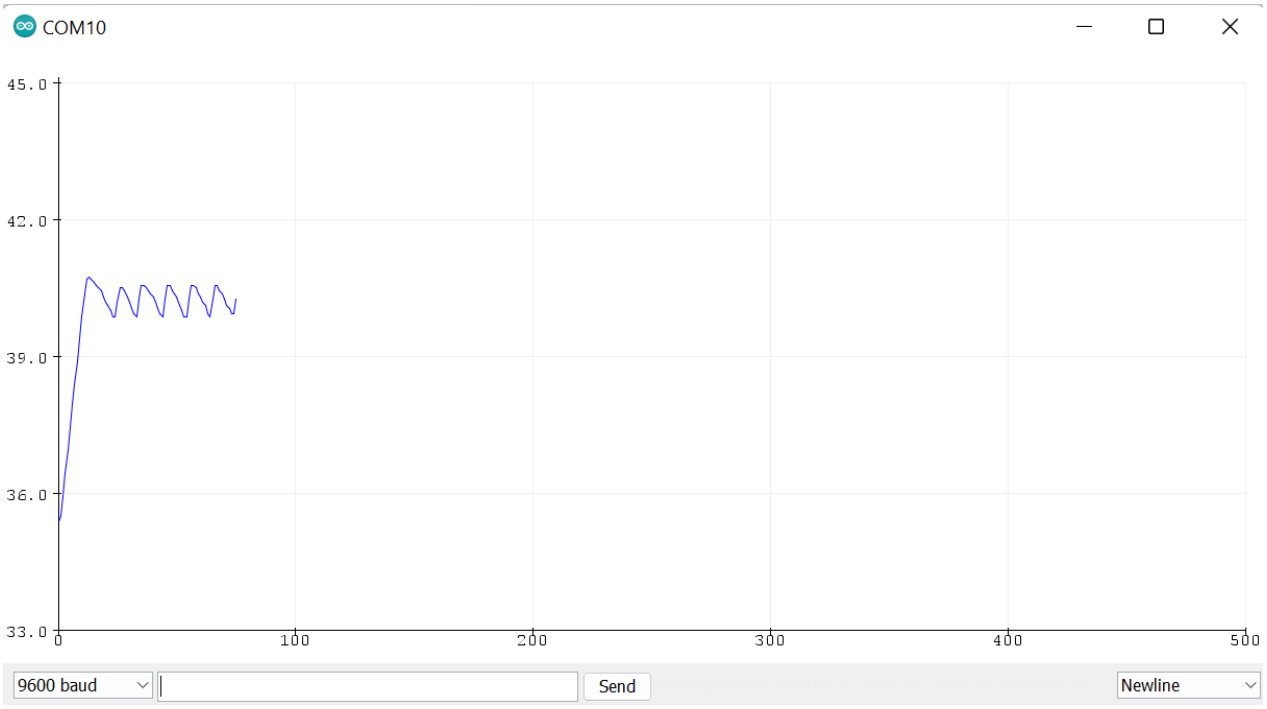
## First trial : $K_d = 0$ , $K_i = 0.5$ , $K_p = 5$



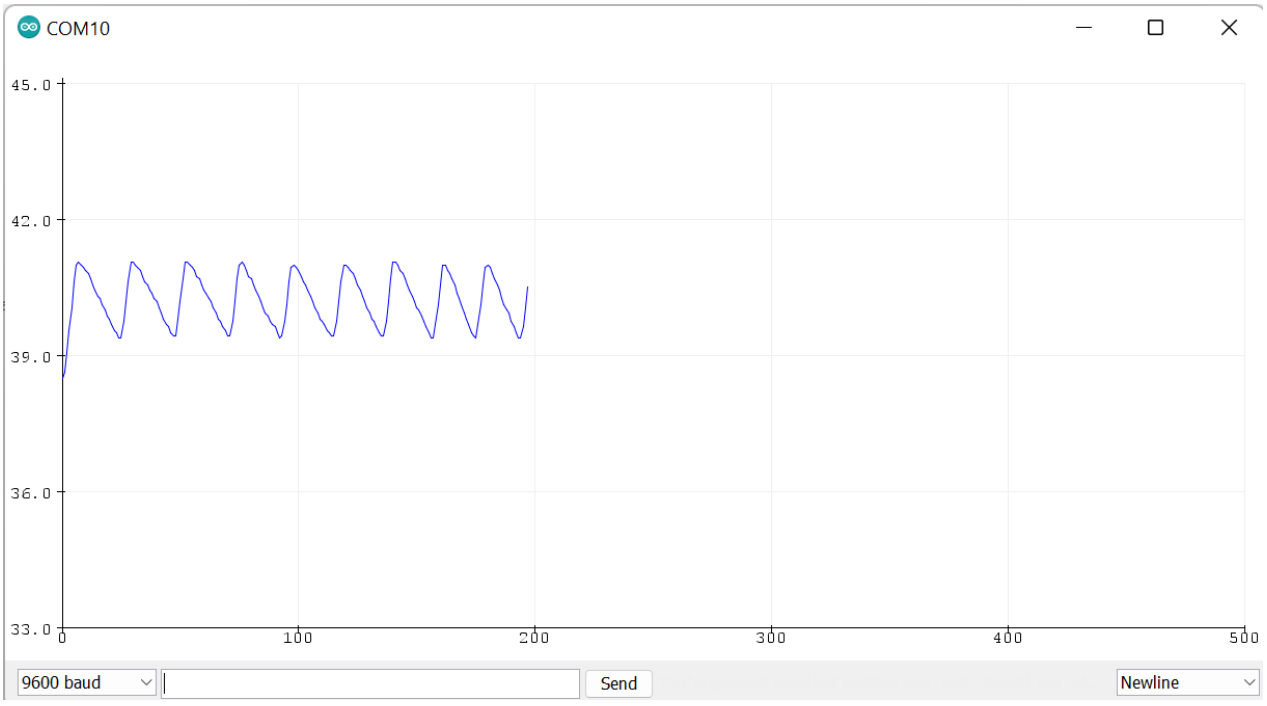

## Second trial : $K_d = 0$ , $K_i = 1$ , $K_p = 5$



**Third trial :** $K_d = 0$ , $K_i = 10$ , $K_p = 15$

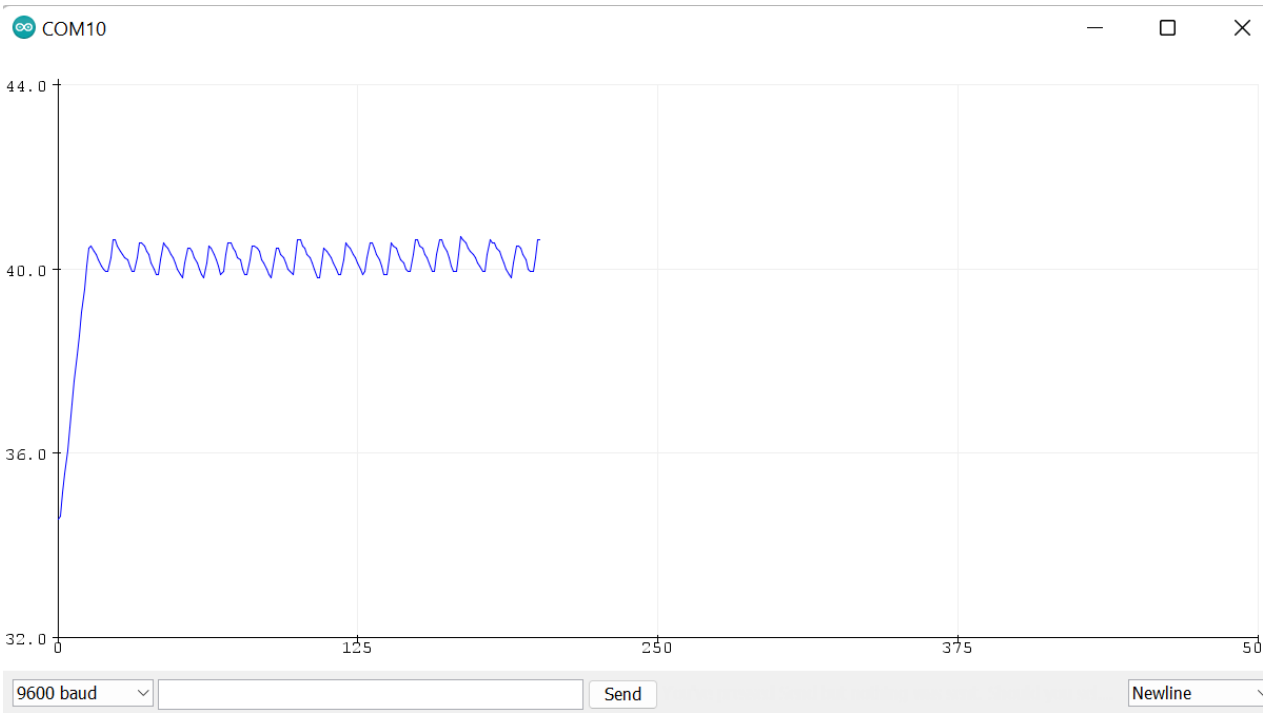



**Fourth trial :** $K_d = 0$ , $K_i = 10$ , $K_p = 1$



# 7.Final Graph

**Final trial :** $K_d = 5$ , $K_i = 15$ , $K_p = 30$



## 8.Complete Code

```
#include <OneWire.h>
#include <DallasTemperature.h>
/********************************************************************/
// Data wire is plugged into pin 2 on the Arduino
#define ONE_WIRE_BUS 10
// Setup a oneWire instance to communicate with any OneWire devices
// (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);
/********************************************************************/
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);
/********************************************************************/
double sensed_output, control_signal;
double setpoint;
double Kp; //proportional gain
double Ki; //integral gain
double Kd; //derivative gain
int T; //sample time in milliseconds
unsigned long last_time;
double total_error, last_error;
int max_control;
int min_control;
int heater = 9;
int fan=8;
int val;
int cel;
void setup() {
  //Serial.println("Dallas Temperature IC Control Library Demo");
 // Start up the library
 sensors.begin();
T = 500;
Kd=0;
Ki = 0.5;
Kp=5;

setpoint = 40;

Serial.begin (9600) ;
pinMode (heater, OUTPUT) ; // set pin as output for relay
digitalWrite (heater, HIGH) ;
pinMode(fan, OUTPUT);
}
```

```
void loop() {

 sensors.requestTemperatures(); // Send the command to get temperature readings

sensed_output=sensors.getTempCByIndex(0);
 //Serial.print("Temperature is: ");
 Serial.println(sensors.getTempCByIndex(0));
 //Serial.print("       "); // Why "byIndex"?

sensed_output = sensors.getTempCByIndex(0);
PID_Control ();
/*Serial.println ("control_signal:");
Serial.print (control_signal);*/
//Serial.print ("sensed_output: ");
//Serial.println (sensed_output) ;
if (control_signal > 0.5) {
digitalWrite (heater, LOW );// turn relay 1 ON
   digitalWrite(fan, HIGH);

}
else if (control_signal < -0.5 ){
digitalWrite (heater, HIGH) ;// turn relay 1 ON
digitalWrite(fan, LOW);

}
delay (100) ;
}

 void PID_Control() {
 unsigned long current_time = millis () ;
 int delta_time = current_time - last_time; //delta time interval
 if (delta_time >= T){
 double error = setpoint - sensed_output;

 total_error += error; //accuma lates the error - integral term
 if (total_error >= max_control) total_error = max_control;
 else if (total_error <= min_control) total_error = min_control;
 double delta_error = error - last_error;
 control_signal = Kp*error + (Ki*T) *total_error + (Kd/T) *delta_error;
 last_error= error;
 last_time = current_time;
 }
 }
```