NATIONAL INSTITUTE OF TECHNOLOGY PATNA, BIHAR

PROJECT REPORT

EE56103 Microprocessor and Microcontroller

# Digital Tachometer using Arduino

By: Abhishek Keshri (2302129)

## Acknowledgement

We would like to express our sincere gratitude to all those who supported and guided us throughout the completion of this project, “Tachometer Using Arduino.”

First and foremost, we are deeply thankful to our professor Dr. Gangandeep Meena for their constant guidance, encouragement, and valuable insights, which helped us successfully complete this work. We also extend our appreciation to our faculty and institution for providing us with the resources and facilities necessary to carry out this project.

We would also like to acknowledge the contributions of team members for their dedication, cooperation, and collaborative effort in every stage of this project — from research and hardware assembly to programming and documentation. Working as a team has not only enhanced our technical knowledge but also improved our problem-solving and teamwork skills.

Lastly, we would like to thank our friends and family for their constant encouragement and support, which motivated us to complete this project successfully.

## Abstract

This project presents the design and implementation of a low-cost digital tachometer using an Arduino microcontroller. A sensor, such as an infrared (IR) sensor, is employed to detect the rotational movement of a motor or shaft by generating pulses corresponding to each revolution. The Arduino processes these pulses, calculates the rotational speed in revolutions per minute (RPM), and displays the readings in real time via a serial monitor or LCD module.

The primary objective of this project is to create an accurate, affordable, and user-friendly instrument for measuring rotational speed. The system demonstrates the practical use of microcontrollers, sensors, and embedded programming in building efficient measurement devices. Its simplicity and adaptability make it suitable for educational purposes, laboratory experiments, and small-scale industrial applications. This project also lays a foundation for further enhancements, such as wireless data transmission, data logging, and integration into advanced control systems, showcasing the potential of microcontroller-based solutions in automation and instrumentation.

**Contents**

## Introduction

A tachometer is an instrument used to measure the rotational speed of a shaft or disk, typically in revolutions per minute (RPM). It is widely used in mechanical, automotive, and industrial systems to monitor the performance of rotating machines such as motors. Accurate measurement of speed is essential for ensuring efficiency, safety, and reliability in these systems. Traditional tachometers are often mechanical or optical in nature, but with the advent of microcontrollers, low-cost digital tachometers can be developed.

This project focuses on the design and implementation of a digital tachometer using an Arduino microcontroller. The Arduino serves as the processing unit that counts the number of pulses generated by an infrared (IR). Each pulse corresponds to one revolution (or a fraction of a revolution, depending on the setup). By counting the pulses over a fixed time interval, the Arduino calculates the RPM and displays it either on a serial monitor or on an external display module.

The digital tachometer offers several advantages compared to conventional methods. It is cost-effective, compact, and relatively simple to construct, making it suitable for laboratory experiments, hobbyist projects, and small-scale industrial applications.

In this project, an Arduino UNO board, a sensor, and optional display modules are used to build a working tachometer prototype. The report explains the working principle, hardware connections, programming logic, and experimental results. The aim is to demonstrate how microcontroller-based systems can be used to design reliable and efficient instruments for real-time measurement of rotational speed placed near the rotating body.

## Objective

1. To design and implement a digital tachometer system based on Arduino and IR sensor.

2. To utilize optical reflection principles for non-contact detection of shaft rotation.

3. Developing an algorithm for Arduino to count pulses and convert them into RPM values.

4. To successfully integrate all the elements and achieve desirable operation from the components.

5. To accurately measure and display motor shaft speed in revolutions per minute (RPM).

6. To provide a clear and user-friendly output through Serial Monitor or LCD display.

## Hardware used:

**1. Arduino UNO**

- **Function**: Arduino UNO is the main microcontroller board used in this project. It processes the pulses from the sensor and calculates the RPM.
- **Working**: It receives digital signals from the sensor through its GPIO pins and executes the programmed instructions to count pulses, calculate RPM, and display the result.

---

**2. IR Sensor**

- **Function**: Detects the rotation of the motor or shaft.
- **Working**: Uses an infrared LED and photodiode. When a reflective surface or object passes in front, the sensor output changes, producing a digital pulse.Each pulse corresponds to one rotation (or part of a rotation).

---

**3. 16x2 LCD (with I2C module)** ***(Optional)***

- **Function**: Displays the measured RPM.
- **Working**: The Arduino sends calculated RPM values to the LCD through I2C communication. The LCD then shows the real-time RPM value to the user. If an LCD is not used, readings can be observed on the Serial Monitor of Arduino IDE.

---

**4. DC Motor / Rotating Shaft**

- **Function**: Acts as the test subject whose RPM is to be measured.
- **Working**: A small DC motor or any rotating shaft is used. A reflective tape or magnet is attached to the shaft so the sensor can detect each revolution.

---

**5. Power Supply**

- **Function**: Provides the required operating voltage to Arduino and other components.
- **Working**:
    - Arduino UNO is powered through USB (5V) or an external adapter (7–12V).
    - The sensor and LCD are powered using the Arduino's 5V pin.
    -

## Circuit diagram:

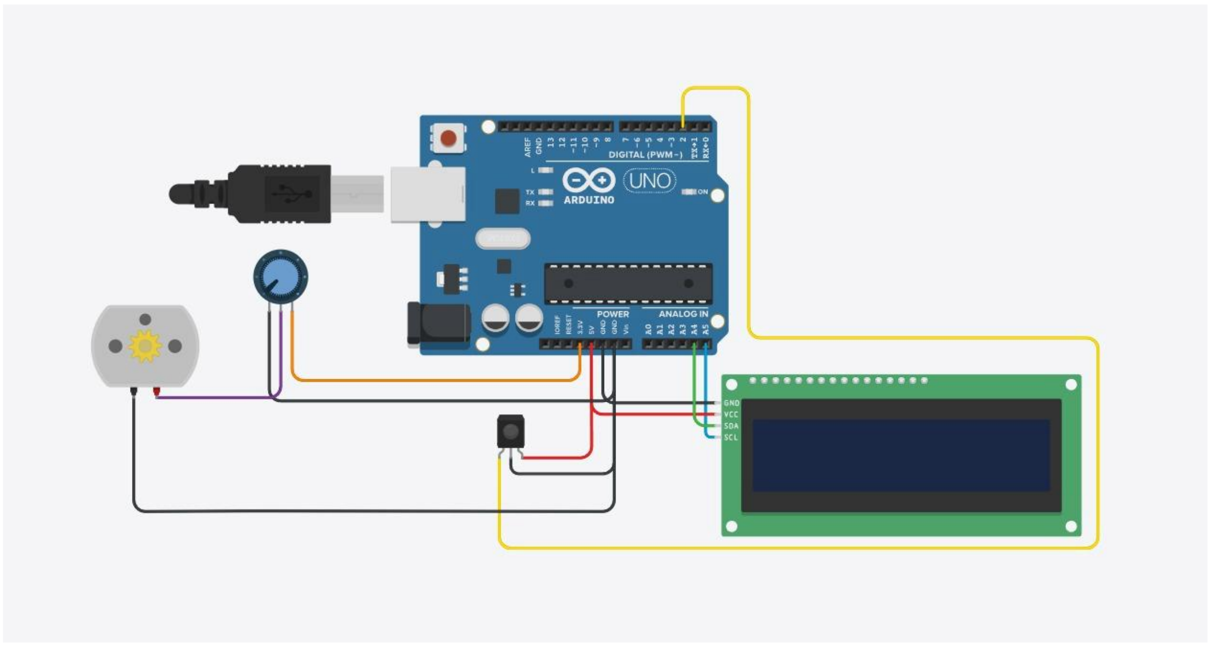


Fig no. 1.1: circuit diagram

## Connections:

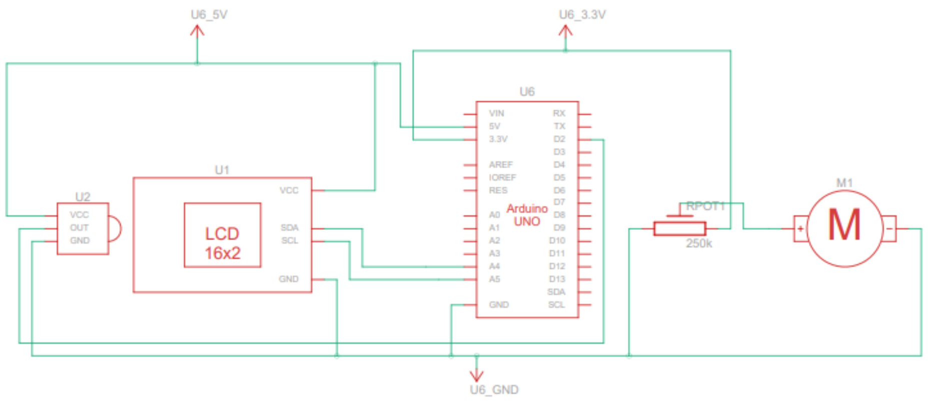


Fig no. 1.2: schematic diagram

## Software used:

- Arduino IDE
- Tinker CAD

## Working Principle

The tachometer works on the principle of converting rotational motion into electrical pulses and then using a microcontroller to process these pulses to calculate the rotational speed in revolutions per minute (RPM).

1. Pulse Generation by Sensor
    - An IR sensor or Hall effect sensor is placed near the rotating shaft or motor.
    - A reflective tape is fixed on the shaft.
    - Every time the reflective tape or magnet passes the sensor, the sensor output changes state (HIGH to LOW or vice versa). This generates one pulse per revolution.

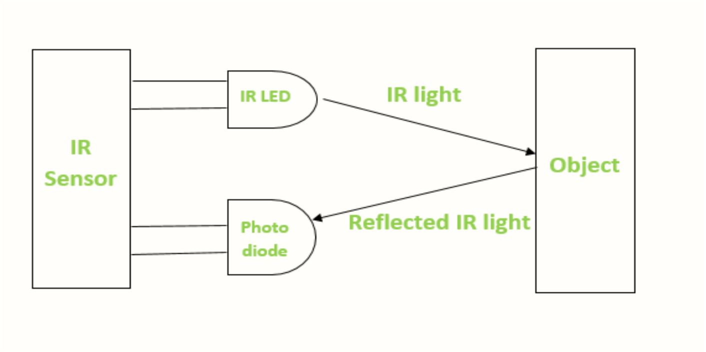


Fig no. 1.3: working of IR sensor

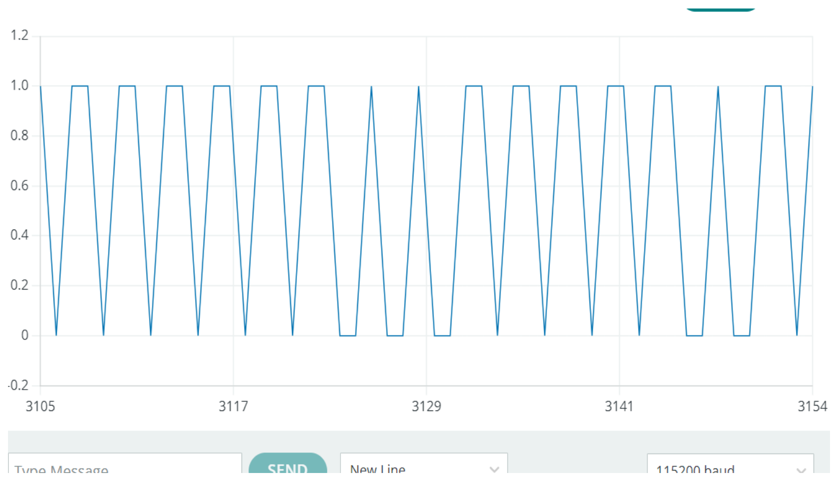


Fig no. 1.4: pulses generated by IR sensor shown on serial plotter in Arduino IDE

2. Pulse Counting by Arduino
   - The sensor output is connected to the Arduino digital input pin.
   - The Arduino counts the number of pulses generated within a fixed time interval (usually 1 second).
   - Since one pulse corresponds to one revolution, the pulse count gives the number of revolutions per second (RPS).
3. RPM Calculation
   - The revolutions per second (RPS) is multiplied by 60 to obtain the revolutions per minute (RPM):

$$\text{rpm} = \frac{\text{count} \times 60.0}{\text{interval\_in\_sec} \times \text{MARKERS}}$$

Where:

Count:no. of pulses in the interval
Interval_in_sec=5 seconds
MARKERS:no. of reflective markings on blade/disc

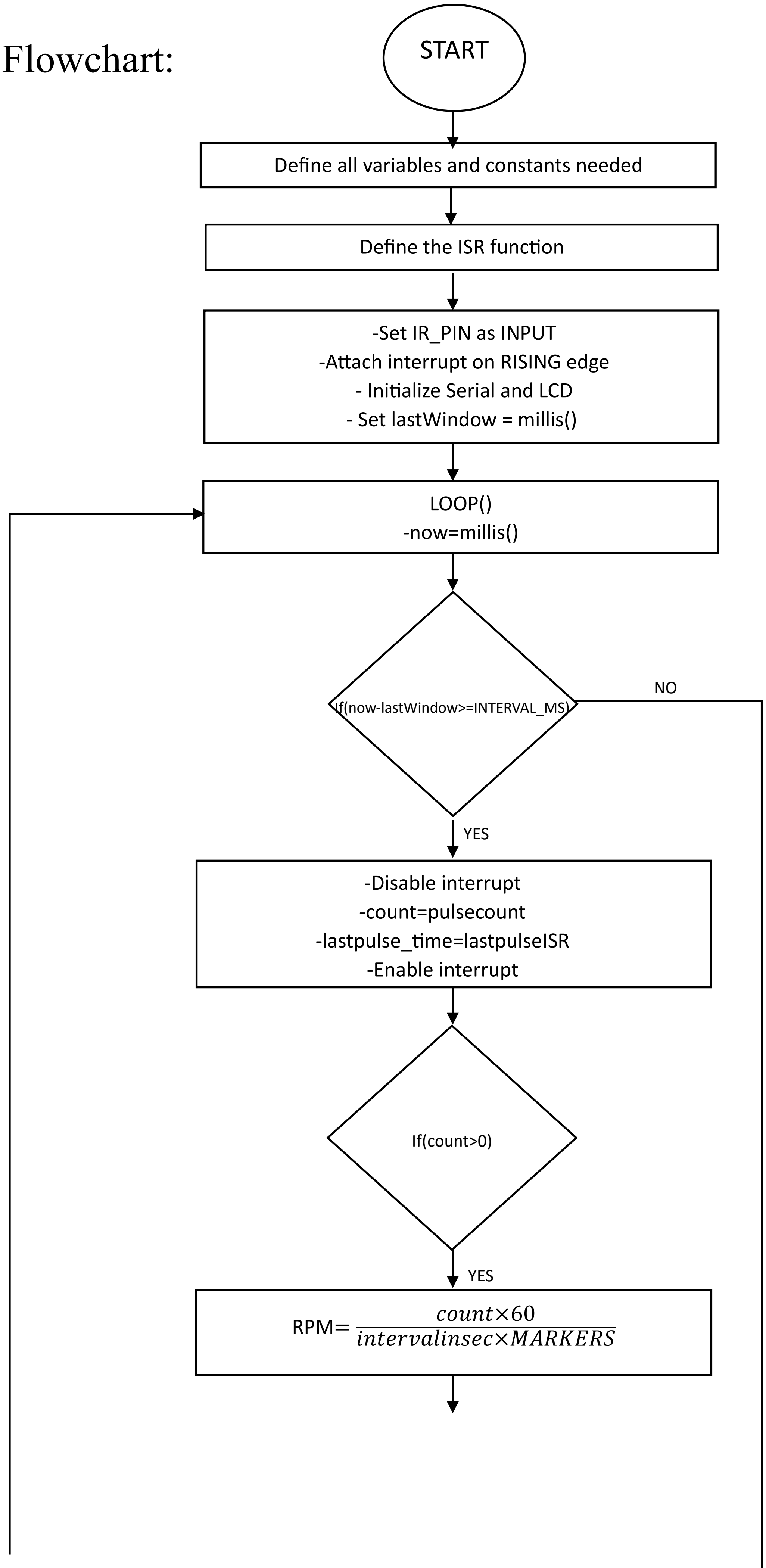
$$\text{corrected}_{\text{RPM}} = (\text{EMA}_{\text{ALPHA}} \cdot \text{rpm}) + \left((1.0 - \text{EMA}_{\text{ALPHA}}) \cdot \text{corrected}_{\text{RPM}}\right)$$

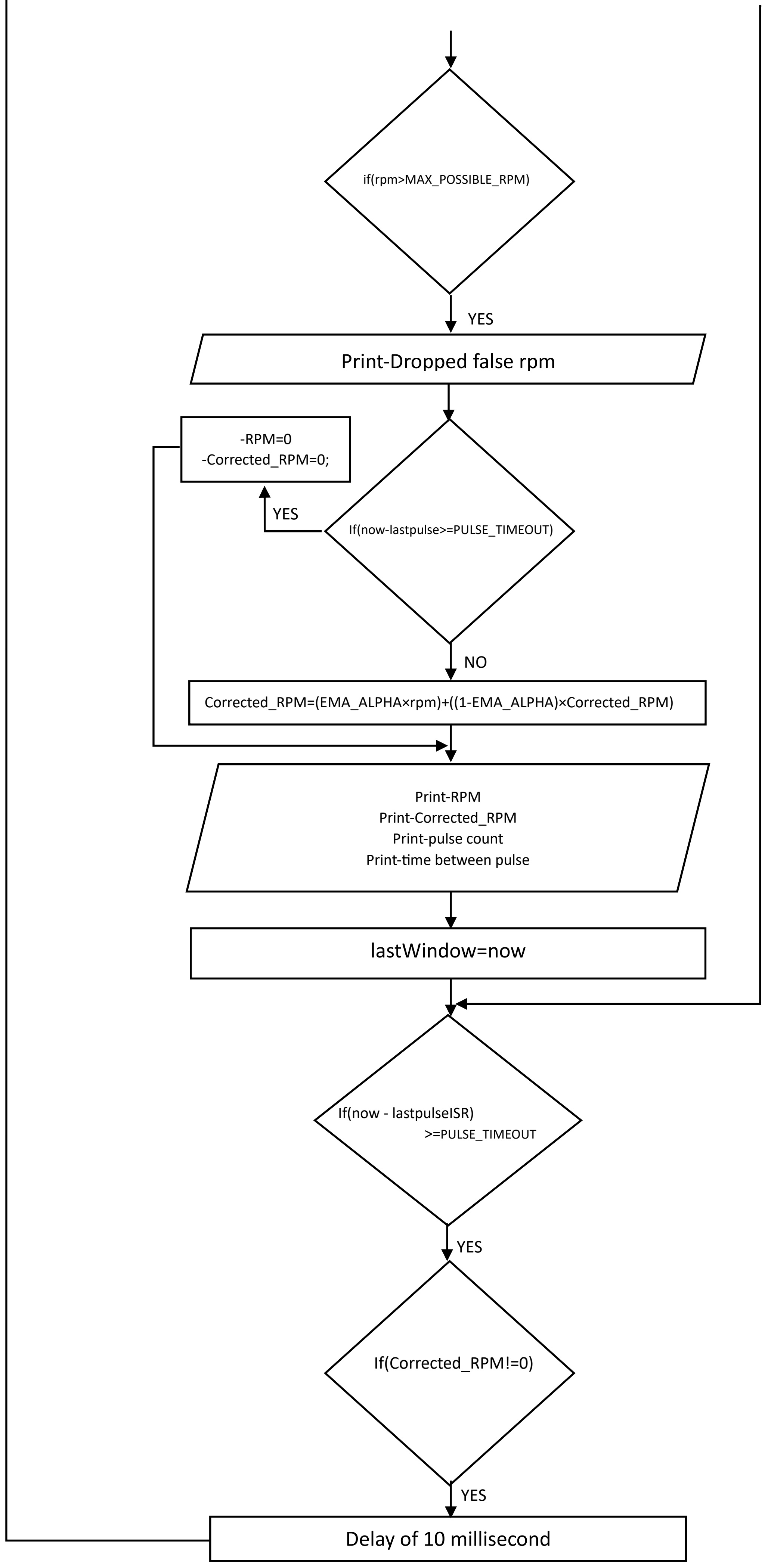
Where:

$\text{EMA}_{\text{ALPHA}}$:Exponential Moving Average constant
rpm: calculated rpm based on Raw data

4. Display of Results
   - The calculated RPM value is displayed either on the Arduino Serial Monitor or on an LCD/OLED display connected to the Arduino.

## Flowchart:


START
Define all variables and constants needed
Define the ISR function
-Set IR_PIN as INPUT
-Attach interrupt on RISING edge
- Initialize Serial and LCD
- Set lastWindow = millis()
LOOP()
-now=millis()
If(now-lastWindow>=INTERVAL_MS)
NO
YES
-Disable interrupt
-count=pulsecount
-lastpulse_time=lastpulseISR
-Enable interrupt
If(count>0)
YES
RPM= $\frac{count \times 60}{intervalinsec \times MARKERS}$


if(rpm>MAX_POSSIBLE_RPM)
YES
Print-Dropped false rpm
-RPM=0
-Corrected_RPM=0;
YES
If(now-lastpulse>=PULSE_TIMEOUT)
NO
Corrected_RPM=(EMA_ALPHA×rpm)+((1-EMA_ALPHA)×Corrected_RPM)
Print-RPM
Print-Corrected_RPM
Print-pulse count
Print-time between pulse
lastWindow=now
If(now - lastpulseISR)
>=PULSE_TIMEOUT
YES
If(Corrected_RPM!=0)
YES
Delay of 10 millisecond

## Programming code:

```
// HW-201 IR Sensor based Tachometer with 5 sec refresh
//Counts pulses (5s window) + I2C LCD + Serial
//Using EMA smoothing to filter noise

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define IR_PIN 2     // out pin for IR sensor
#define INTERVAL_MS 5000UL // counting window (5 seconds)
#define MARKERS 1// no. of reflective marks on rotating blade/disc

#define MIN_PULSE_INTERVAL_US 1000UL // time gap between two valid signals
#define PULSE_TIMEOUT 6000UL // 6sec threshold to determine motor has stopped
#define MAX_POSSIBLE_RPM 20000UL //To discard very high results due to errors
    //can change depending on device rpm  being measured

/*EMA(Exponential Moving Average),a techniue used to filter noise
  in sensor data by adjusting the weight of current and previous
  reading to calculate the result*/

const float EMA_ALPHA = 0.25;

//volatile keyword to avoid caching the values in registers
volatile unsigned long pulseCount = 0;
volatile unsigned long lastpulse = 0; //for finding time difference
volatile unsigned long lastpulseISR = 0;//for timeout

unsigned long lastWindow = 0;//track time to calculate last RPM value
float corrected_RPM = 0.0;// final value of RPM

LiquidCrystal_I2C lcd(0x27, 16, 2); //Initialising the LCD

// ISR function using rising edge trigerring
void  on_pulse() {
//micros to calculate time in milliseconds
  unsigned long current_time = micros();
//to filter noise in pulse
  if (current_time - lastpulse >= MIN_PULSE_INTERVAL_US)
   {
    pulseCount++;
    lastpulse = current_time;
    lastpulseISR = millis();
  }
}

void setup() {
  pinMode(IR_PIN, INPUT);

  //executing the ISR and picking the pin for execution
  attachInterrupt(digitalPinToInterrupt(IR_PIN), on_pulse, RISING);
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Tachometer Ready");
  delay(700);
  lastWindow = millis();
  corrected_RPM = 0.0;
}

void loop() {
  unsigned long now = millis();//capture start time

  //check for timing threshold for calculation
  if (now - lastWindow >= INTERVAL_MS) {
    noInterrupts();
   //diasable interrupt to export values into local variables
    unsigned long count = pulseCount;
    pulseCount = 0;
    unsigned long lastpulse_time = lastpulseISR;
    interrupts();//enable after successful export

    //convert from miiliseconds to seconds
    float intervalinsec = INTERVAL_MS / 1000.0;
    float rpm = 0.0;

    //formula for calculating RAW_RPM
    if (count > 0) {
      rpm = (count * 60.0) / (intervalinsec * (float)MARKERS);

    //drop results with errors
    if (rpm > (float)MAX_POSSIBLE_RPM) {
        Serial.print("Dropped false RPM: ");
        Serial.println(rpm);
        rpm = 0.0;
      }
    }

    //timeout check
    if ((now - lastpulse_time) >=PULSE_TIMEOUT) {
      rpm = 0.0;
      corrected_RPM = 0.0;
    } else {
      //to remove noise due to change in sensor position or environment
      corrected_RPM   =   (EMA_ALPHA   *   rpm)   +   ((1.0   -   EMA_ALPHA)   *
corrected_RPM);
    }

    // Serial output
    Serial.print("Count:");
    Serial.print(count);
    Serial.print("  RawRPM:");
    Serial.print(rpm, 1);
    Serial.print("  corrected_RPM:");
    Serial.println(corrected_RPM, 1);

    // LCD output
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("RPM:");
    lcd.setCursor(5, 0);
    lcd.print((int)round(corrected_RPM));
    lcd.setCursor(0, 1);
    lcd.print("P:");//pulse count in the interval
    lcd.print(count);
    lcd.print(" T:");//time since last pulse
    lcd.print(((now - lastpulse_time) < 9999) ? (now - lastpulse_time) :
9999);

    //store last iteration value
    lastWindow = now;
  }

  //show 0 if no pulses were received beyond timeout
  if ((now - lastpulseISR) >= PULSE_TIMEOUT) {
    if (corrected_RPM != 0.0) {
      corrected_RPM = 0.0;
      Serial.println("No pulses -> forced RPM=0");

    }
  }
//delay to avoid overload of buffer on screen and serial monitor
  delay(10);
}
```

## Result:

During testing, the RPM of the DC motor was observed to be **approximately 760 to 780 RPM**, with minor fluctuations due to motor speed variations and sensor sensitivity. The results demonstrate that the tachometer is capable of providing reliable and reasonably accurate speed measurements for small motors and low- to medium-speed rotating systems.

## Future Scope:

1. **Wireless Connectivity**
   - Integration with Wi-Fi or Bluetooth modules (ESP32, HC-05) to transmit RPM data wirelessly to smartphones, PCs, or IoT dashboards.
2. **Data Logging and Analysis**
   - Addition of SD card modules or cloud storage services for storing long-term speed data for analysis and performance tracking.
3. **Industrial-Grade Accuracy**
   - Use of higher-precision optical or laser sensors for better accuracy in high-speed or industrial applications.
4. **Compact PCB Design**
   - Designing a custom printed circuit board (PCB) for a smaller, more robust, and production-ready tachometer unit.
5. **Mobile App Integration**
   - Development of a companion app to display RPM readings in real-time, generate reports, and send alerts.
6. **Automated Control Systems**
   - Integration with motor controllers or automation systems to dynamically adjust motor speed based on feedback from the tachometer.

## Conclusion

This project successfully demonstrates the design and implementation of a digital tachometer using an Arduino microcontroller. By integrating a low-cost sensor with Arduino, the system effectively measures the rotational speed of a motor or shaft and displays the RPM in real-time. The use of simple electronic components, such as an IR sensor or Hall effect sensor, along with a display module or serial interface, makes the system both affordable and easy to replicate.

The project highlights the versatility of microcontroller-based systems in creating accurate and efficient measurement instruments. It provides hands-on experience with sensor interfacing, pulse counting, and embedded programming, making it an excellent educational tool for students and hobbyists.

While the current design is suitable for small-scale applications, it can be further enhanced with features like wireless data transmission, data logging, higher accuracy sensors, and integration into industrial monitoring systems. Overall, this tachometer offers a reliable, cost-effective solution for measuring rotational speed and serves as a foundation for more advanced instrumentation and control projects.

## References

- https://medium.com/illumination/arduino-with-infrared-sensor-48ad4415f320
- https://arduino.stackexchange.com/questions/11907/how-do-i-even-out-infared-distance-sensor-readings
- https://projecthub.arduino.cc/mircemk/arduino-tachometer-rpm-meter-with-ir-sensor-module-a36d7c
- https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/