FAST-National University of Computer and Emerging Sciences Islamabad.

Project: Development of Spin Coater for Printed Electronic Device Fabrication

Group Members: Kamal Ud Din(13i-0474)

Muhammad Asif(12i-0460)

Mudassir Abbass(12i-0408)

Supervisor: Dr. Shawkat Ali

Department: Electrical Engineering

Developers' Submission:

This report is being submitted to the Department of Electrical Engineering of the National University of Computer and Emerging Sciences in partial fulfillment of the requirements for the degree of BS in Electrical Engineering.

Developer's Declaration:

"We take full responsibility of the project work conducted during the Final Year Project (FYP) titled "***Development of Spin Coater for Printed Electronic Device Fabrication***". We solemnly declare that the project work presented in the FYP report is done solely by us with no significant help from any other person; however, small help wherever taken is duly acknowledged. We have also written the complete FYP report by ourselves. Moreover, we have not presented this FYP (or substantially similar project work) or any part of the thesis previously to any other degree awarding institution within Pakistan or abroad.

We understand that the management of Department of Electrical Engineering of National University of Computer and Emerging Sciences has a zero-tolerance policy towards plagiarism. Therefore, we as an author of the above-mentioned FYP report solemnly declare that no portion of our report has been plagiarized and any material used in the report from other sources is properly referenced. Moreover, the report does not contain any literal citing of more than 70 words (total) even by giving a reference unless we have obtained the written permission of the publisher to do so. Furthermore, the work presented in the report is our own work and we have positively cited the related work of the other projects by clearly differentiating our work from their relevant work.

We further understand that if we are found guilty of any form of plagiarism in our FYP report even after our graduation, the University reserves the right to withdraw our BS degree. Moreover, the University will also have the right to publish our names on its website that keeps a record of the students who committed plagiarism in their FYP reports."

Kamal Ud Din	Muhammad Asif	Mudassir Abbass
BS (EE) 13i-0474	BS (EE) 13i-0460	BS (EE) 13i-0408

Certified by Supervisor

Verified by Plagiarism Cell Officer

Dated: _____

Abstract:

The demand of the present time is to innovate such devices and component which are small in size, efficient in performance and cost. With the advancement in mobile technologies and IoT the size of the circuit matters a great deals. Printed electronics is the emerging field where the world's research is getting the focus at. Spin coater is a device which is efficient to make a coat of chemical liquid substance on a substrate and produces thin film circuits, this process is called spin coating.

The commercially available Spin coating devices are priced in range of $2000-2500. Due to lack of awareness and resources most the technical institutes in Pakistan have not been able to introduce their students with the printed electronics technology; hence very small scale research is going on in this field which is emerging and promises to be backbone of future technologies. Making an efficient machine which could provide research opportunities to the younger generation of Pakistan serves as our problem statement. Furthermore, to show the coating quality we produce a coated Wi-Fi antenna by using our own developed Spin Coater. The antenna is coated with Nano-particles of gold chemical ink.

The results of this project are successful because we produce an affordable and economical Spin coater device with a proposed model. Furthermore, a wearable antenna is fabricated through spin coating which shows the ease and strength of printed electronic devices.

Acknowledgements:

We would like to express our greatest appreciation to all those who provided us with all the necessary help regarding our Final Year Project. We would like to express our deepest gratitude to out Project Supervisor, “Dr. Shawkat Ali” for trusting in us, and investing his full efforts in guiding us to complete our project.

We would like to appreciate the suggestions given by other faculty members, as well as our concerned panel, whose part in stimulating recommendations and encouragements, helped us in achieving our goal. Furthermore, we would like to acknowledge with much recognition, the most important role of Lab Supervisor, “Engr. Kashif Islam” who provided us all the needed equipment and guided us about purchasing required modules.

Table of Contents:

Chapter 1 (Introduction)

1.1 Background Information

1.1.1 Printed Electronics

Printed electronics is an all-encompassing term for the printing method used to create electronic devices by printing on a variety of substrates. Originally, printed electronics related to organic or plastic electronics that use one or more inks made of carbon-based compounds. As demand for wearable devices and thinner electronics expands, printed electronics are being used to form flexible keyboards, antennas, electronic skin patches, and more. Printed electronics technology has evolved over time, and now inkjet printers are capable of printing electrical circuits quite inexpensively and quickly. In short, printed electronics is one of the fastest growing technologies today and is becoming invaluable to several industries including healthcare, aerospace, media, and transit [6].

Printed electronics have become secure, flexible, and cost-effective, all of which make them appealing to a broad range of industries. Printed circuitry has the potential to reduce costs and technical constraints typically associated with mass producing electronics. Printed electronics also require fewer input materials and less energy to work with them. And, printed electronics pave the way for flexible devices that people previously may not have thought possible. For example, companies are working on using printed electronics for identifying banknotes, credit cards, legal documents, and other items with unique printed signatures. Another industry benefiting from printed electronics advancements is photovoltaics. Printed electronics have the potential to significantly change solar power projects, thanks to less expensive polymer electronics.

- Overall, the benefits of printed electronics include
- Low cost
- Attractive and flexible form factor
- Ease of production

- Ease of integration

Facilitating widespread development of non-conventional functional electronic devices including flexible displays, smart labels, animated posters, active clothing, and more [6].

1.1.2 Spin Coating Process

Spin coater performs Spin coating which is a method to produce thin organic films that are uniform over large areas. Usually a small amount of coating material is applied on the center of the substrate, which is either spinning at low speed or not spinning at all. The substrate is then rotated at high speed in order to spread the coating material by centrifugal force. Rotation is continued while the fluid spins off the edges of the substrate, until the desired thickness of the film is achieved [1].

Spin coating process can be broken down into the four stages.

I. Deposition of the coating fluid onto a flat substrate.

II. The substrate is accelerated up to its final, desired rotation speed.

III. The substrate is spinning at a constant rate and fluid viscous forces dominate the fluid thinning behavior.

IV. The substrate is spinning at a constant rate and solvent evaporation dominates the coating thinning behavior. After evaporation of the whole solvent, a solid film is generated [2].

The figure-1.2.1 given below shows the spin coating process.

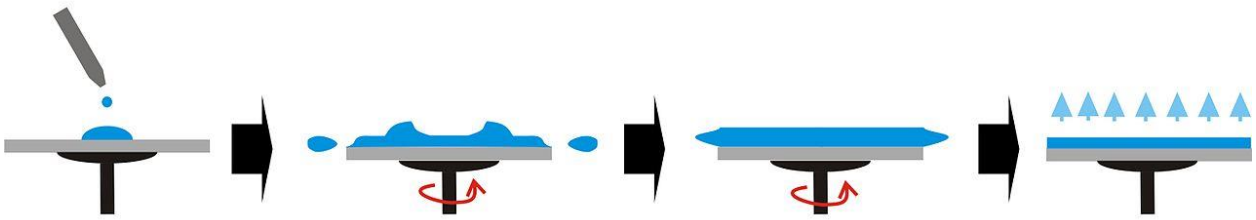


Figure-1.2.1

1.1.3 Wearable Printed Antenna

Mobile devices are ruling the current electronic market as communication is becoming essential part of daily life. The communication devices demand a system that is easy to carry, energy efficient

and fast in performance to meet the demands. Wearable printed antennas are getting a huge attention because they are paper thin in size, more reliable, cost effective and fast as they provides high speed data connectivity.

There are various method to make the thin layer of coating on the substrate surface to form these wearable antennas. The approaches can be multiple to get the desired results such as screen printing, flexography, offset lithography, inkjet printing and spin coating. Each method provides a different advantage keeping the design of the antenna in prospect. As our project yields a functional spin coater as a product so we have preferred spin coating technique to do an attempt to make the antenna as spin coating is much simple and efficient technique.

1.2 Problem and Solution

1.2.1 Problem Statement

Our goal is development of Spin Coater as a product which is used for electronic components fabrication that is easy to use, economical to buy, efficient in performance and accessible for researchers. Our scientific society lacks the culture of developing electronic components fabrication tools. In result, firstly, electronic components are imported into the country or the required equipment is imported at a high cost. The equipment such as spin coater has numerous applications in semiconductor industry and research labs of material sciences to provide the researchers opportunity of fabricating customize products or components which are required.

1.2.2 Our Proposed Model

Our proposed model is developed keeping researchers and students in prospect. Availability of user friendly and economical device would improve the efficiency of research and save time. The proposed model results in development of low power and stable machine which has negligible oscillations.

The existing models of spin coater are costly and not user friendly. We propose a digitalized spin coater, which has microcontroller based precise controlling unit, input keypad and LCD display unit. Moreover, commercially available spin coater without rotary pump (Model: Kw-4A) has price around $1200 (PKR 124800), which is very expensive. Whereas our proposed spin coater costs comparatively much less which estimates to be around (PKR 15000) only, which is approximately 90% cost effective than the commercial one. Moreover, there is no time consuming delay and it is easy to clean after each operation.

A high RPM brushless DC motor is mounted on firm wooden surface with the help of screws. The extended part of the motor rotor is attached to a circular plastic disk. The disk is flat and uniformly shaped. To carry out the spin coating process the disk will hold a substrate in the center. To maintain the substrate on the disk the existing models of spin coaters use suction pumps whereas in our model we will use screw lock and double tape to fix the substrate on the rotating disk. A keypad will be employed to provide desired inputs for controlling the equipment operations, speed and time. A motor driver circuit is used to control the rotation of motor. Microcontroller is used to

control the plant and carryout all operations and run the motor with the help of motor driver circuit. The motor is capable of achieving speed up-to 11500 RPMs on 12 V DC power supply input. Pulse Width Modulation (PWM) technique is used in this process to control the speed of motor, where the microcontroller generates controlled signal with duty cycles [7]. The PWM signal is sent to motor driver to vary the voltage supply to motor to generate a ramp and then maintain a constant speed. The existing controlling techniques do not implement Proportional Integration Derivative (PID) technique to control the motor, whereas we are implementing PID technique. The implementation of PID is effective to get a quick response of the system and minimize errors which results in minimizing oscillations of the system.

Chapter 2 (Solution/Design/Implementation)

2.1 Block Diagram

The fig-2.1 shows block diagram of the spin coater, and describes all the modules connected to the microprocessor. The user can enter the ramp RPM, steady state RPM and time into the system. The LCD will display the input and output information and the update in case of down count and RPM rise. Motor driver IC is a current amplifier; the function of motor driver is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor. The IR sensor provides the feedback loop in the system to bring accuracy in the desired and obtained RPMs. The rotating substrate is part of rotor of the motor and it also effects the RPMs and requirement of power to obtain the desired RPMs.

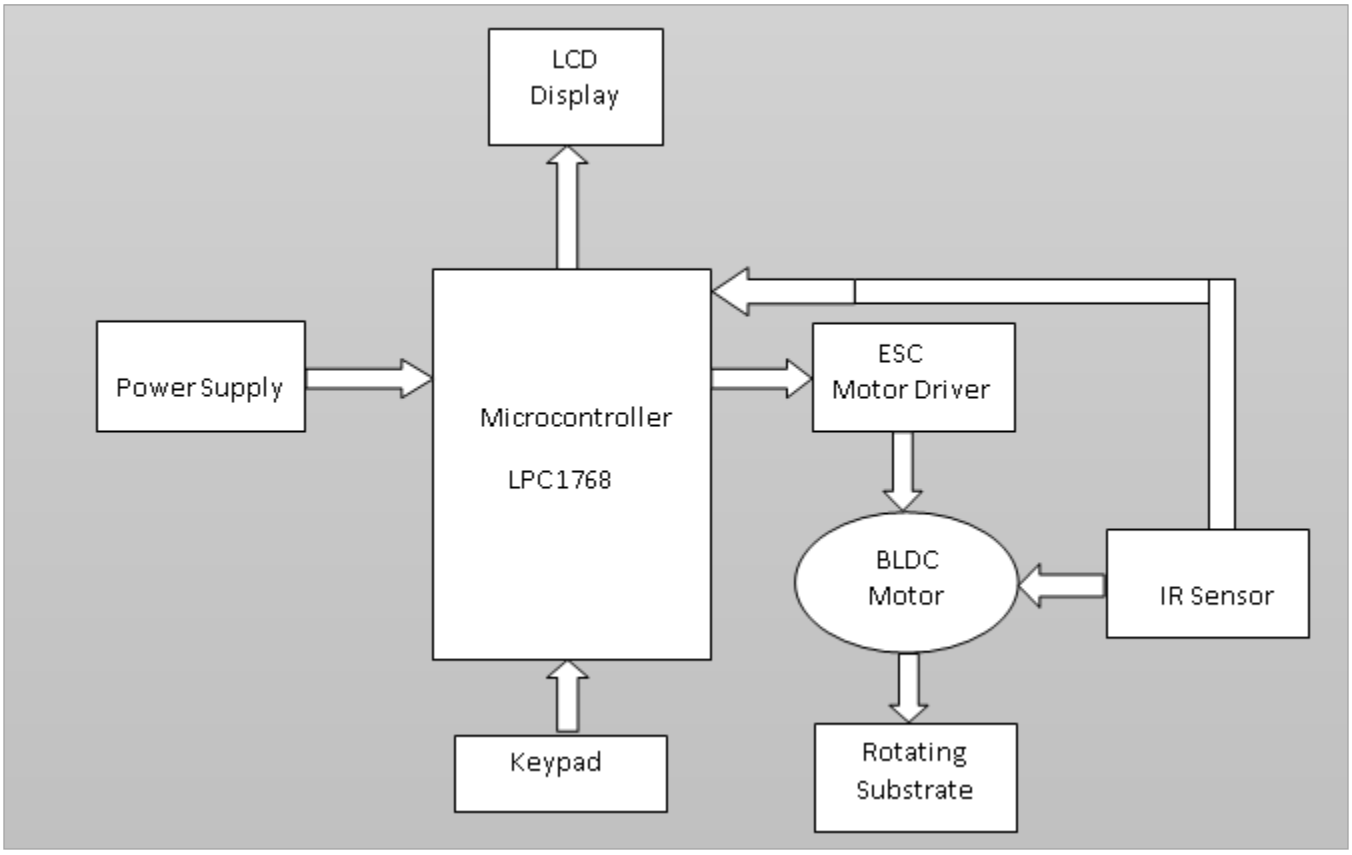


Fig-2.1

2.2 Module Specifications

Following are the details of all the modules of the spin coater.

1) Microprocessor

The microcontroller takes the input data from keypad and generates a proportional PWM signal for the PWM controlled brushless DC motor. The ongoing status is shown on LCD. LPC 1768 is shown in (fig-2.2.1).

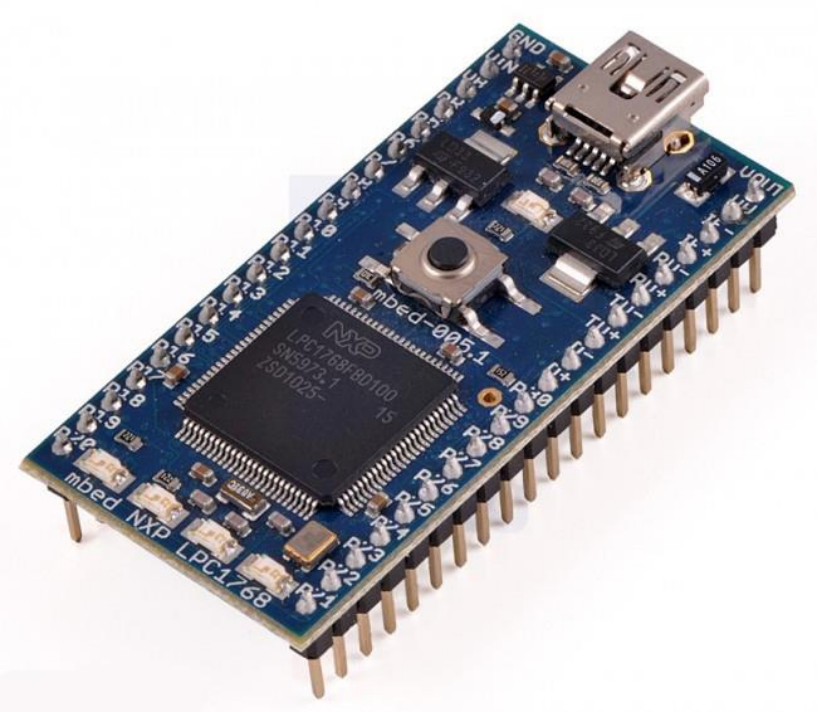


Fig-2.2.1

a) Data Input:

Data from the keypad gets corresponding result in BLDC motor speed range: 0 rpm to 12,000 RPM

b) Data Output:

Pulse Width Modulated (PWM) signal gives PWM duty cycle: 0 % to 100%.

2) Keypad

It is 4x4 numeric keypad. (fig-2.2.2)

a) Key Pressure:

2N to 3N

b) Interface:

USB and RS232

c) Input power Supply:

5Vdc @ max 100mA

Fig-2.2.2

3) LCD

The specifications of LCD (fig-2.2.3) are given below:

Character LCD 16x4

a) 5x8 dots includes cursor

b) Built-in controller (ST7066)

c) +5V power supply

d) 1/16 duty cycle

e) LED can be driven by PIN1, PIN2, PIN15, PIN16 or A and K

f) Interface : 6800, option SPI/I2C (RW1063 IC)

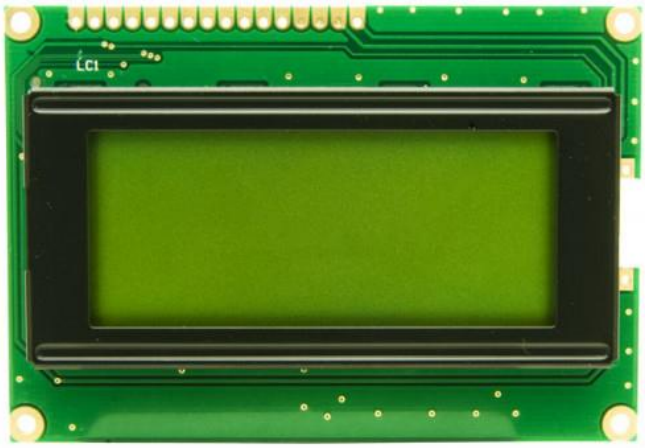


Fig-2.2.3

4) *Motor Driver IC (ESC):*

The ESC (fig-2.2.4) which was used in the project has following features:

a) Data Input:

Voltage signal (PWM) of Range: 0V to 5V

b) Data Output

Voltage signal of Range: 8V to 16V and current signal of range 0 to 3 A.

Fig.2.2.4

5) Brushless DC Motor

The BLDC motor is shown in fig-2.2.5.

a) Input:

Voltage signal of Range: 8V to 16V and current signal of range 0 to 3 A.

b) Output:

Motor RPM of Range: 0 to 11500.

Fig.2.2.5

6) Infra-red Sensor

. The IR sensor is shown in fig-2.2.6. This part of the system will separately check the RPMs of the motor to determine the errors by comparing the current rotation value with the desired rotation value.

Fig-2.2.6.

7) *Rotating Substrate*

The rotating motor will get the substrate to rotate with the speed of the motor as it is mounted on it. The size and nature of material of substrate depends on the desired product properties. A glass substrate is shown in figure-2.2.7.

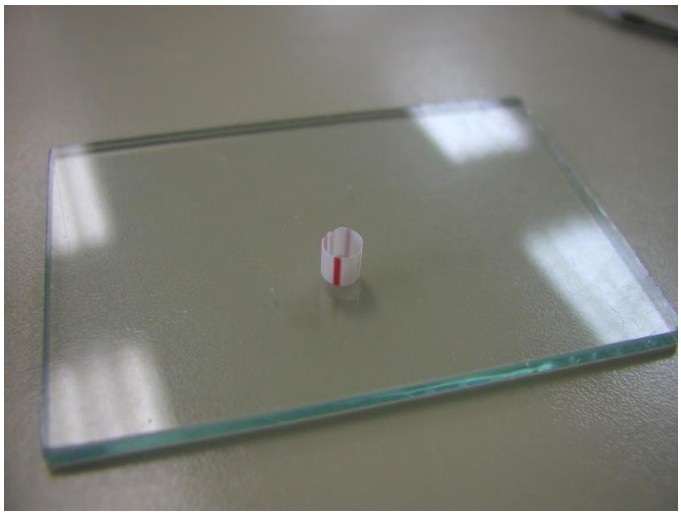


Fig-2.2.7.

2.3 Flow Chart

The flowchart in fig-2.3 describes detail process of the system working.

2.3.1 Description of Flow Chart Blocks

The flow chart explains the working logic of the model. As we turn on the switch the system is ready to take input from the user. The required input are both RPM and Time for ramp and Steady state stages. If this data input is correct then the system will move to the next step and in case if it is incomplete then it will come back to the start. The LCD will display the ramp and steady state

RPMS and a down counter will count until zero then the motor rotation will stop and that is when our coating process completes. If an error occurs during this process the system stops and takes the user to the starting point again.

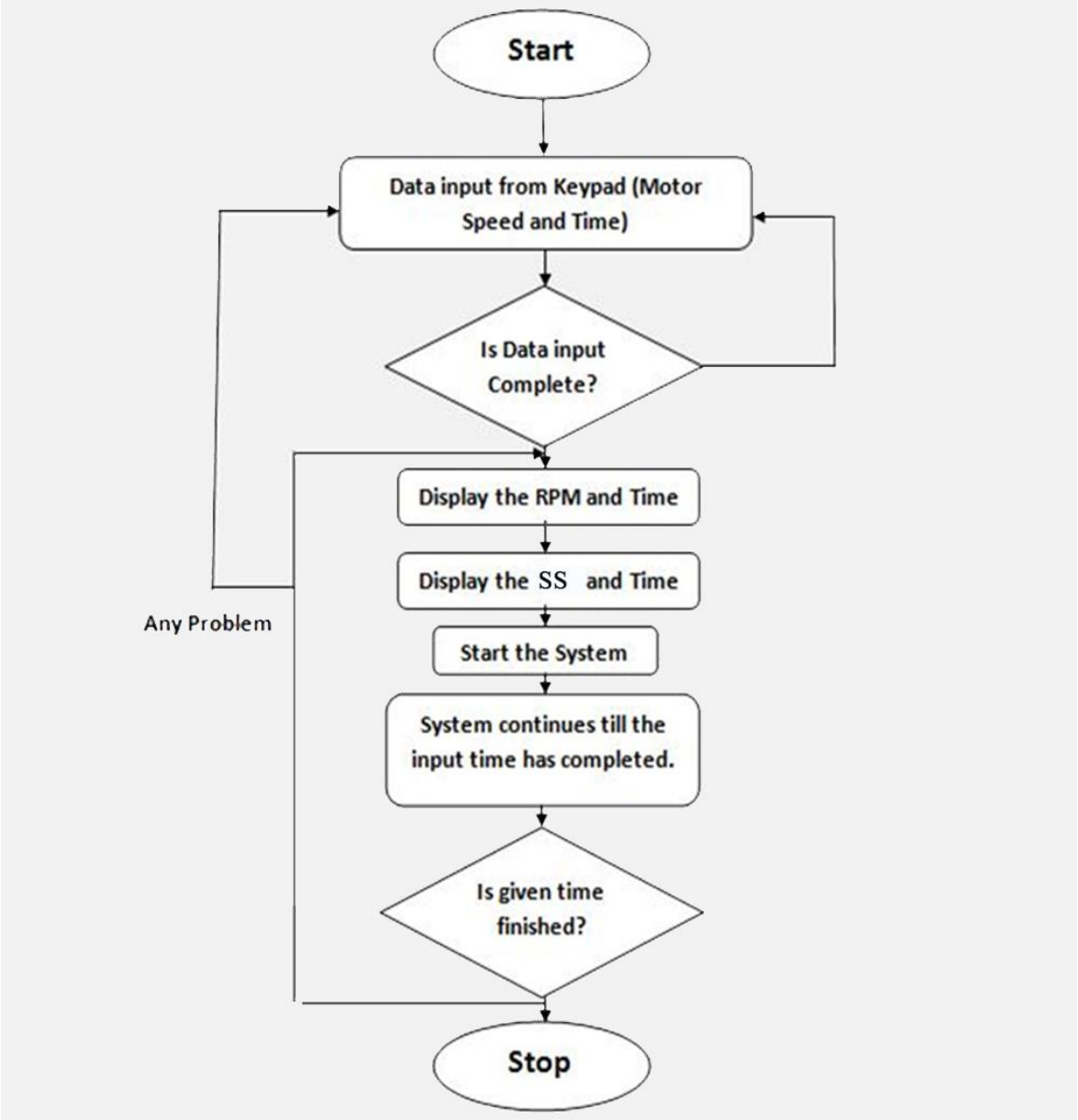


Fig-2.3

2.4 System Interfacing and Design

2.4.1 LCD Interfacing:

We interface 4x16 LCD with microcontroller using following pin connection as mentioned in Figure-2.4.1.1. Data bus of LCD is 8-bit and other three pins RS, R/W, E are for register select, read/write and enable respectively. Potentiometer of 10k is for LCD brightness, so user can adjust its brightness through this variable resistor. LCD is powered by microcontroller.

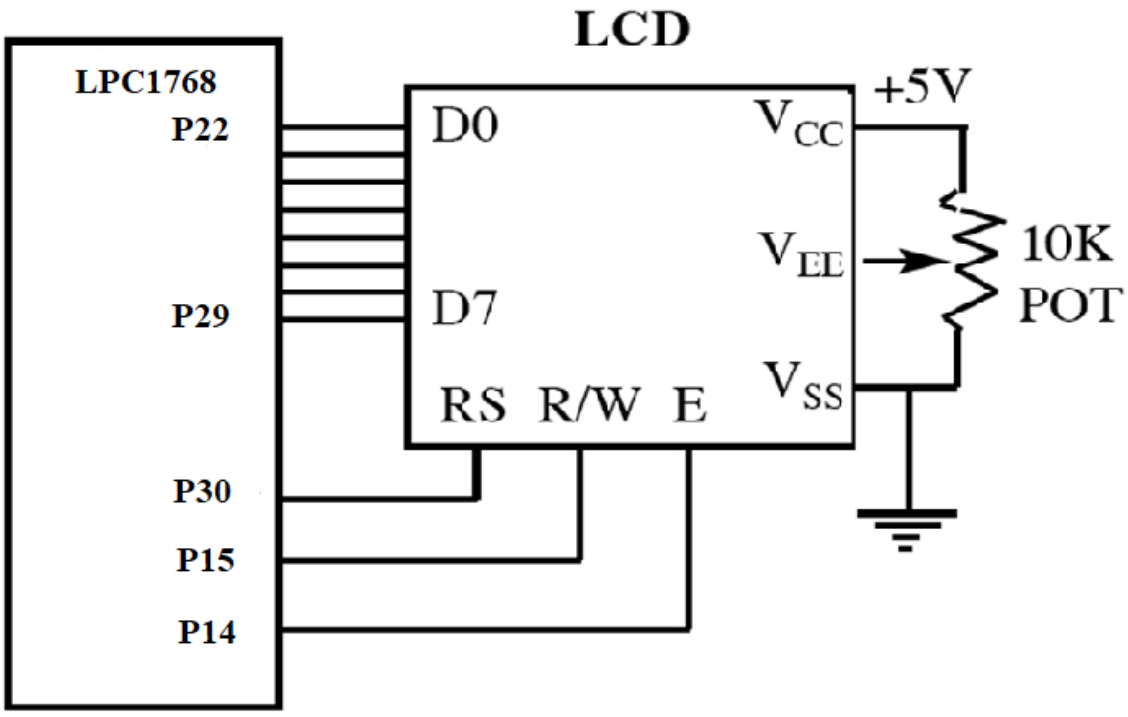


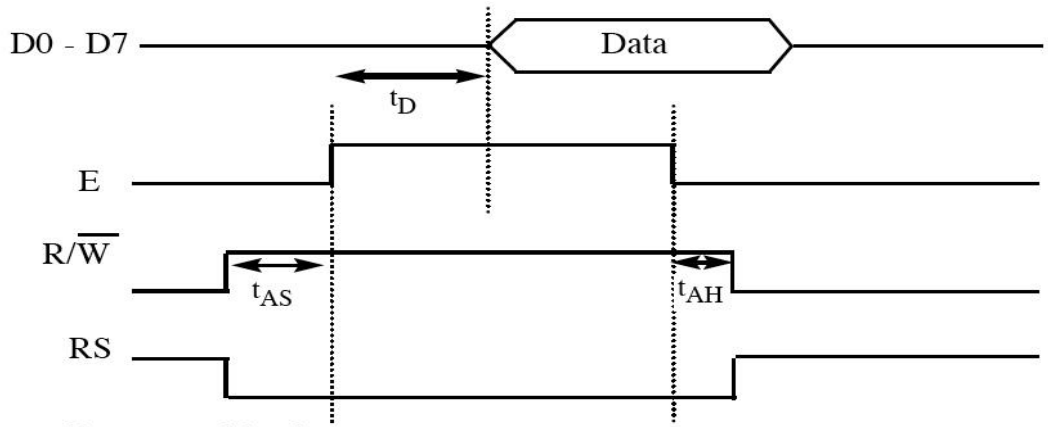
Figure-2.4.1.1: LCD pin connection with microcontroller LPC1768

After applying connections it's necessary to understand the working of LCD through timing diagram. We write its code according to the timing diagram which is given in Figure-2.4.1.2 [2].

RS register is used to select a register either it can be command register or data register. Before display of data user should send some appropriate commands to set the LCD functionality according to its requirements. That is why first of all select command register by sending 0 and then data register by sending 1 on RS pin.

R/W allows us to read or write from LCD, by sending 0 it will allow to write and by sending 1 it will allow to read.

Enable pin is used to enable LCD. When data is supplied enable pin should go from high to low as shown in Figure-2.4.1.1.



t_D = Data output delay time
t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

Note: Read requires an L-to-H pulse for the E pin.

Figure-2.4.1.2: LCD Timing for Read (L-to-H for E line)

2.4.1.1 Steps for Command Write

These are the following steps for command write,

1- Apply 0 to enable pin E.
2- Put command on data pins. There are different commands as shown in Table-2.4.1. Use of commands depends on require functionality of LCD.
3- Send 0 to RS, because we are sending command and it will select command register in LCD.
4- Send 0 to R/W, because we are writing command on LCD.
5- Send High to Low pulse on E pin for 0.5 microseconds.
6- Put a delay of 100 microseconds between two consecutive commands.

2.4.1.1 Steps for Data Write

Following are the steps for Data write:

1- Apply 0 to enable pin E
2- Put data on data pins which you want to display on LCD.
3- Send 1 to RS, because we are sending data and it will select data register in LCD.
4- Send 0 to R/W, because we are writing data on LCD.
5- Send High to Low pulse on E pin for 0.5 microseconds.
6- Put a delay of 100 microseconds between two consecutive data.

We can send one byte (8-bit) data at a time.

Code (Hex)	Command to LCD Instruction Register
1	Clear Display Screen
2	Return home
4	Decrement Cursor (Shift cursor to left)
6	Increment Cursor (Shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor on
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift entire display to the left
1C	Shift entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 Lines and 5 x 7 matrix

Table-2.4.1: LCD Command Table [2]

This command table provides flexibility to get the functionality of LCD according to the requirements. Usually before sending any data on LCD to display we clear the LCD and turn on the cursor. We can move the cursor to any require position.

2.4.2 Keypad Interfacing:

4x4 Keypad is just like a matrix of 4x4 push buttons as shown in Figure-2.4.2. Columns must be connected to Vcc and rows to ground. When user pushes any one of the key then the connection between that specific row and column is established that is why that specific column give 0 at output. Microcontroller scans all keys in a specific manner to detect that pressed key. All four rows and columns are connected with microcontroller. Pull up the resistor of connected pins of microcontroller with columns, as we mentioned above is that columns must be connected with high logic [3].

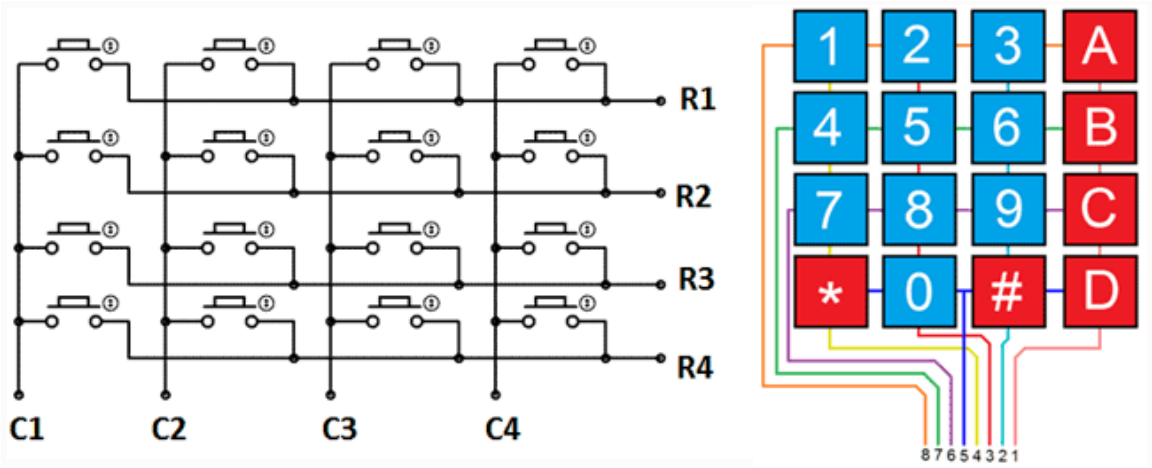


Figure-2.4.2: 4x4 Matrix Keypad and its internal circuitry

2.4.2.1 Key detection process

These are the following steps for key detection,

1- Assume that all keys are released and there is no single connection between row and column. Microcontroller will read all columns and if all columns are giving logic high than it means that there is no pressed key. Whenever the user will press the key the connection will be established and that column will be grounded. Let's assume that user has pressed R2, C2 key; controller will get this type of sequence 1011 for columns and controller does not know that which row is connected with C2. Now controller has judged that key has pressed because sequence is not 1111.

2- Now controller will find the row that is connected with C2. First of all it will give 0111 to rows and output from columns will be 1111. Then it will give 1011 to rows and output from columns will be 1011, this was a logic sequence that controller was finding. We get that sequence (1011) by setting only second row zero, so pressed key is R2, C2 (key digit 5).

Controller will set ASCII number of digit 5 that is 53. We set ASCII because we want to print this number on LCD and it takes digits in ASCII.

2.4.2.2 Switch Debouncing

When user presses a key it does not go suddenly from low to high or high to low. It goes up and down and becomes stable after very short amount of time as you can see from the figure-2.4.2.2 given below [4]. The switch debouncing time varies for different types of switches. When controller reads a debouncing it understands that key is pressed two or more times or takes garbage value for that pressed key because system processing is too fast than switching debouncing time.

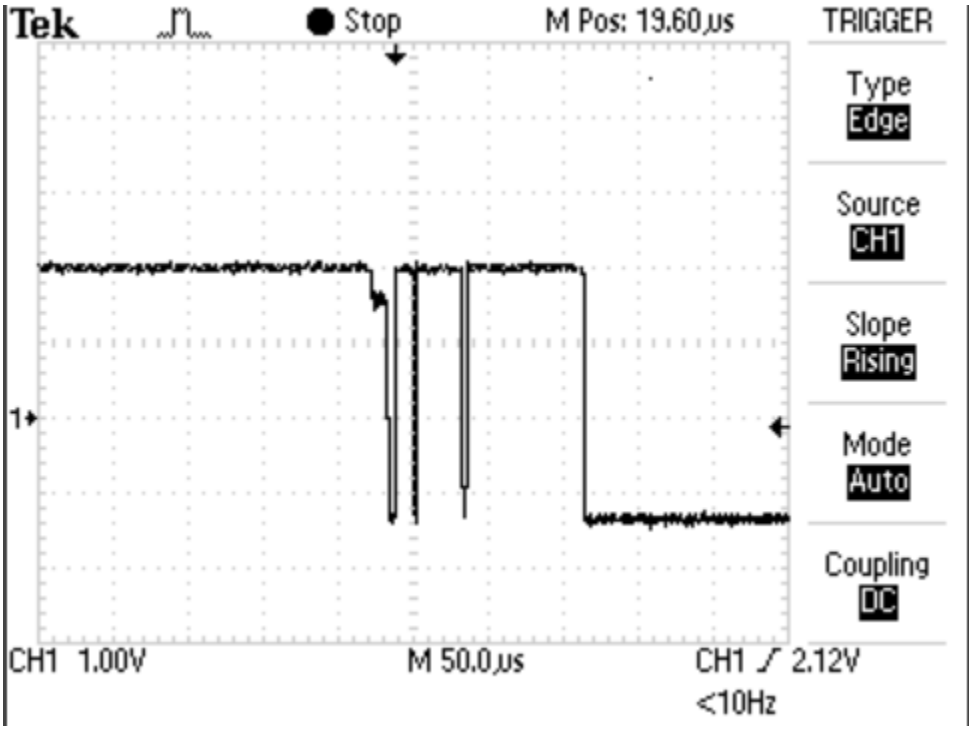


Figure-2.4.2.2 Switch Debouncing when key is pressed

2.4.2.3 Switch Debouncing Solution

There are different solution to remove the switch debouncing, one way is to use D flip-flop to put delay in switch output. But in case of keypad it's very inefficient to use above described technique. We just simply put some adjustable delay to skip switch debouncing, that adjustable amount of delay is 20 microseconds.

2.4.3 System Design

2.4.3.1 Pulse Width Modulation

PWM is a method to generate analogue signal using DC power source. It is used to run brushless DC motor and other electrical devices. It consists of two man things, first one is duty cycle and second one is frequency. Duty cycle of a PWM signal defines that for how much time in a cycle signal is ON or HIGH. We generally represent duty cycle in percentage e. g 25%, 50%, 100% duty cycle [5]. Minimum duty cycle is 0% and maximum is 50%. If a signal is on for half and off for half time in each cycle than duty cycle will be 50% as shown in figure-2.4.3.1.

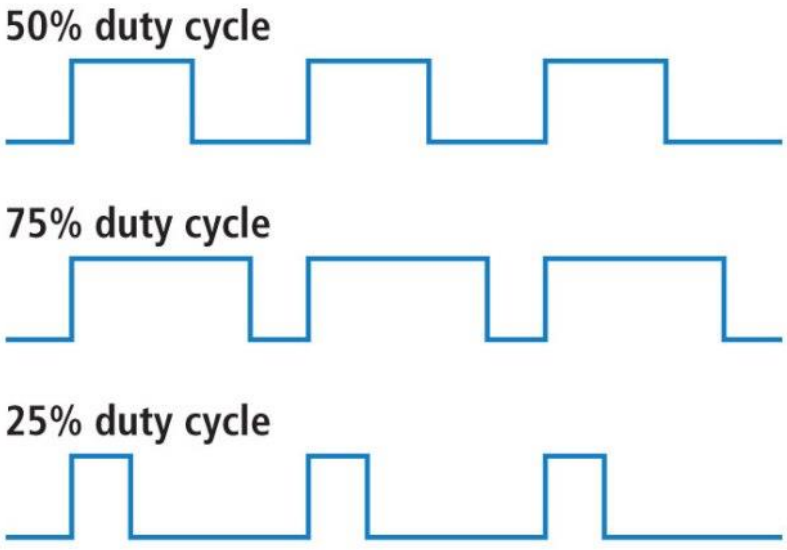


Figure-2.4.3.1: PWM Signals of different duty cycles

If we are applying 25% duty cycle PWM signal to a device and high level of wave has amplitude of 5V, than we can find the analogue voltage at which that device is running.

Duty Cycle = 25% = 0.25

Amplitude of high level of signal = 5V

Analogue voltage at which device is running = 5 * 0.25 = 1.25V

1.25 Volt is a value at which device is running by applying 25% duty cycle PWM. In our case the amplitude of high level of signal is 12V, because it is rated value of brushless DC motor.

2.4.3.2 Electronic Speed Control (ESC)

ESC is electronic device which is used to run brushless DC motors. The basic purpose of this device is to run the motor at its rated value. Output voltage at the pins of LPC-1768 microcontroller is 3.3 volt and the maximum amplitude of PWM signal will be 3.3 volt and current is 0.3 A. This signal cannot run the motor because it's not according to its rated values. According to the rated value of motor maximum voltage should be 12V and minimum current should be 3A. To meet these requirements we need to add some extra circuitry, that extra circuitry is ESC. ESC will take PWM signal from microcontroller and DC power supply according to the rated value and then it will amplify PWM signal according to its duty cycle.

2.4.3.3 RPM Counter Circuit:

To count RPM of Spin coater we use infra-red (IR) sensor. IR sensor's consists of two main components: transmitter and receiver. The transmitter is just like IR LED (Light Emitting Diode), it emits infra-red rays and these rays are invisible through naked eye [6]. IR rays reflect back in presence of obstacle as shown in figure-2.4.3.3, and receiver receives these rays. The intensity of rays depends on two main things,

1- Distance: How for is the obstacle from transmitter and receiver.
2- Surface properties of obstacle. If the surface is black than rays will not reflect back or do with very low intensity. On different colors its behavior is different. The best way to use these sensors effectively is to use black and white surface, because these are contrast colors and detection will be easy. That's why we use these two colors on the lower side of rotating disc.

When rays reflect back on receiver then internal circuitry of receiver becomes active and current starts flowing through receiver. The magnitude of current flowing through the receiver depends on the intensity of rays. We can define this relationship as given below;

Resistance is inversely proportional to rays intensity and current is directly proportional to rays intensity.

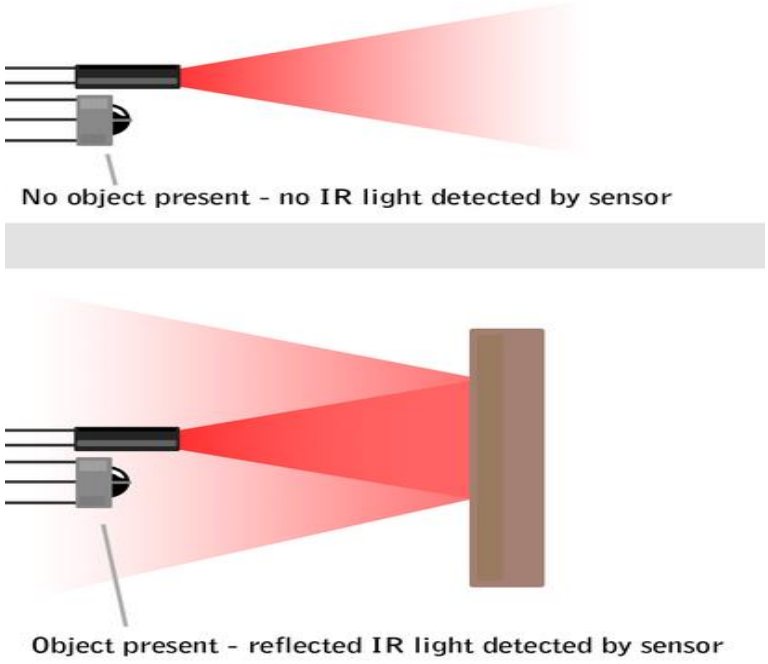


Figure-2.4.3.3: Working of IR Sensor

2.4.3.4 System Interrupts

Interrupt is a signal asserted by hardware or software to processor and indicates that event has occurred and need immediate service from processor [3]. Interrupts can be any internal event or external event. Timers of microcontroller asserts interrupt this is internal interrupt that indicate that timer has completed his counts. If a sensor is attached to your microcontroller as interrupt, then this sort of interrupt indicate that external event has occurred and need some service from processor, this interrupt is called external interrupt [3].

In our system there are total four interrupts,

1- IR sensor interrupt (External Interrupt)
2- Timer1 interrupt for ramp stage (Internal Interrupt)
3- Timer1 interrupt for steady state stage (Internal Interrupt)
4- System reset interrupt (External Interrupt)

IR sensor is counting RPM of motor; this interrupt will occur multiple times in each second during spin coating process. Timer1 interrupt for ramp stage will assert request for service after each second, same case for steady state timer1 interrupt. In case of any damage or wrong values of RPM or time entered by user, system will be ready to accept a reset interrupt from user. Reset interrupt can occur at any time.

When system has multiple interrupts then more than one interrupts can occur at the same time. To solve this problem it's necessary to do interrupt scheduling for system.

- **Interrupts and Shared Data Problem:**

Share data problem occur when a high priority task changes the data before completion of previous task, and previous task can take wrong decisions [2]. In this project IR sensor ISR is counting motors RPM and its interrupt can occur at any time while on other hand Timer1 interrupt is using the data counted by IR sensor ISR and will occur after each second. Timer1 ISR has instructions of PID calculation based on the sensor value, so here is data sharing. We cannot miss any sensor value because our PID value will not be accurate while on the other hand we must have to perform PID after each second otherwise system performance will not be satisfactory. This problem can be solved by doing atomic coding and we did that.

- **Atomic Coding:**

Code that cannot be interrupted by anything that might modify the data being used is called atomic code [4]. In our case IR sensor interrupt is interrupting the PID section. To solve this problem we set the priority of interrupts. In these two interrupts Timer1 interrupt has higher priority than IR sensor interrupt. By applying this problem PID calculations will be accurate and there will be no shared data problem. Whenever these two interrupts will occur at the same time system will provide service to timer interrupts and timer ISR will compute PID and give specific duty cycle signal to motor and will display RPM on LCD. After completion of this, system will provide service to

sensor interrupt. The concept we implemented on our system is describe by the below given diagram.

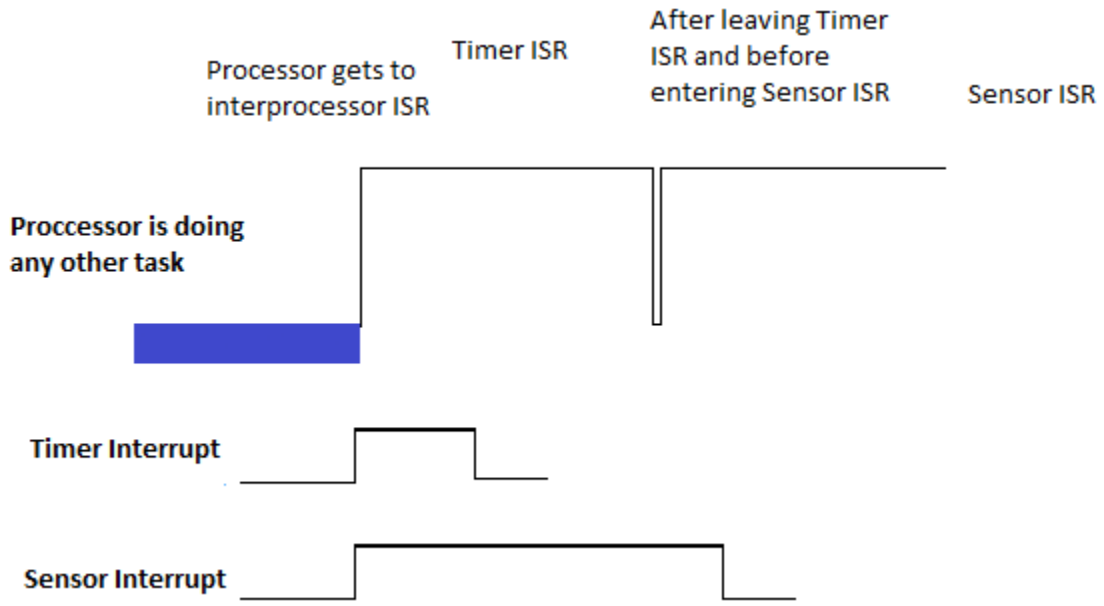


Figure-2.4.3.4 : Processor Execution of Sensor and Timer interrupts with atomic coding

The blue part of diagram indicates that processor is doing any other task and suddenly two interrupts comes at a same time. The processor will provide immediate service to higher priority interrupt and in this case timer interrupt has higher priority. Processor will follow these steps to provide service.

1- It will complete the current instruction execution and will put the address of next instruction (PC) on stack.
2- Program Counter (PC) will be loaded by ISR start address
3- After completion of ISR PC will pick the address from stack and will start its normal routine execution.

In above figure-2.4.3.4 as we see that after the execution of timer ISR it will go back and check that is there any interrupt and if answer will be ‘yes’ than it will provide service to sensor interrupt. After the execution of sensor interrupt PC will load back the instruction address from stack and will perform its previous blue color task.

- **Interrupt Scheduling:**

When system has multiple interrupt then more than one interrupt can occur at same time. System will handle these interrupts by its default technique according to its hardware support. This can manipulate different calculations and expected performance will be compromised. Spin coater has four interrupts as mentioned above. We did their static scheduling to make the system more reliable and accurate with its calculations and execution.

J1 represents job1 which is reset interrupt.

J2 represents job2 which is sensor interrupt.

J3 represents job3 which is timer interrupt for ramp stage.

J4 represents job4 which is timer interrupt for steady state stage.

Process Name	Priority Number
J1	1
J2	4
J3	2
J4	3

Table 2.4.3.4: Priority table for processes

Ramp and steady state timers will not be active at a same time. Ramp timer will be active first and then steady state. Scheduling diagram is given below.

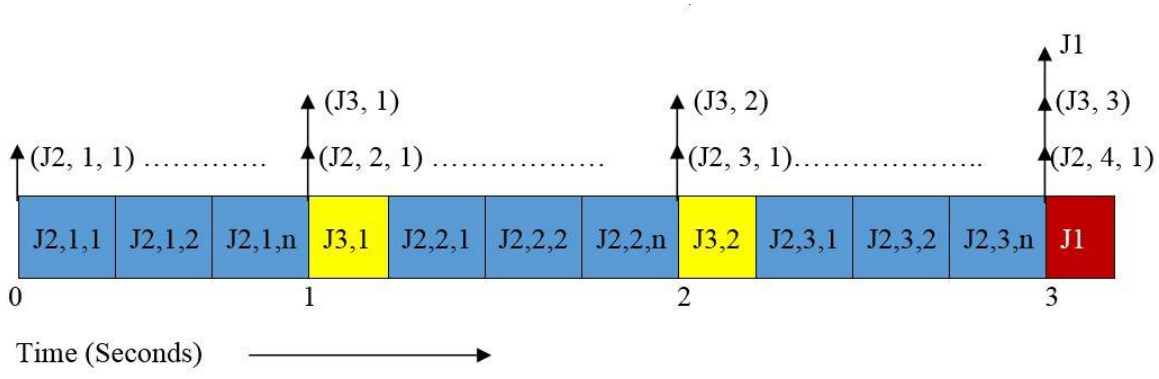


Figure-2.4.3.5 Scheduling diagram considering worst scenario

In this diagram, we consider worst scenario for scheduling. First of all spin coater will run ramp stage and after each one second timer interrupt will interfere. At the start system will provide service to sensor and after one second we consider a worst scenario where both jobs are requesting for service. System will provide service to high priority job first which is J3 and then it will go for J2. J2 and J1 are aperiodic it can interfere at any time, while J3 is periodic with period 1 second. At 3 second we consider this point as worst case where three interrupts are asking for service. Here high priority job is J1 which will get service and this is reset that's why system will reset itself and then it will follow the same scenario. In case of steady state everything will be according to the above given scheduling except J3. Steady state will awake J4 instead of J3.

2.4.4 Motor Control

2.4.4.1 Open Loop Implementation

First of all, we implement open loop on system. In this case we did not use the RPM counter sensor output and try to control the motor using adjustable duty cycle value. The difficult thing in this task was to calculate duty cycle value for every reference value. We derive a general formula by assuming that motor behavior is linear,

At 50% duty cycle the motor speed is 8000 RPM

Duty Cycle per RPM = 50 / 8000 = 0.00625%

Duty Cycle for Reference RPM = 0.00625 x Reference RPM

Let assume that Reference RPM = 5000

Duty Cycle for 5000 RPM = 0.00625 x 5000 = 31.25 %

We implement this concept and got this sort of result as shown in figure-2.4.4.1.

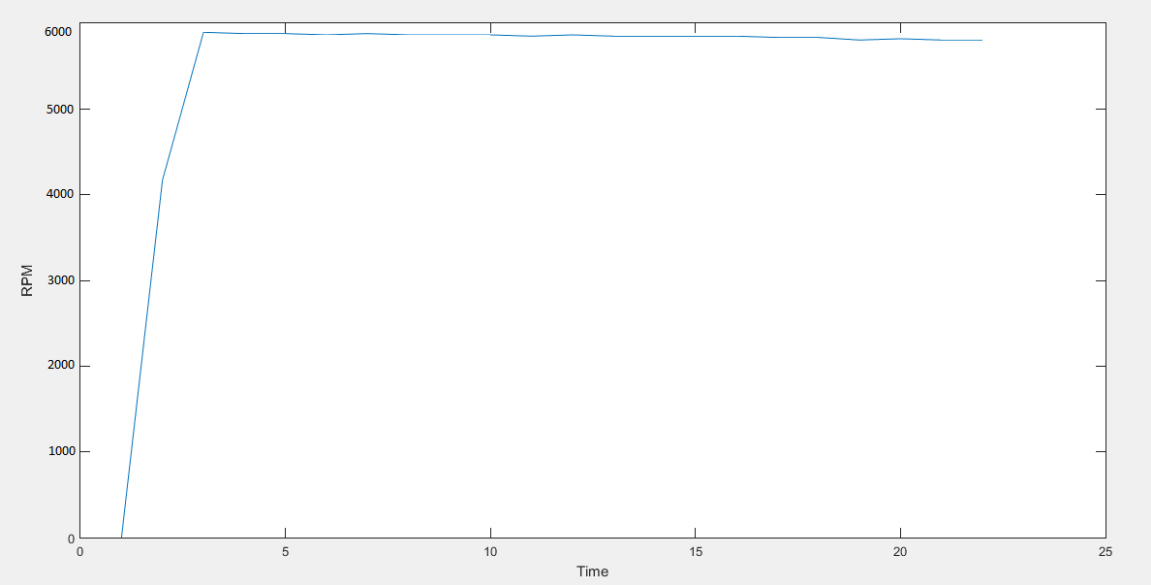


Figure 2.4.4.1: System behavior using Closed Loop at Reference RPM = 4000

As we can see that there is near about 2000 RPM difference as compared to reference value so this is not going to be a desirable system.

2.4.4.2 Close Loop Implementation

To implement closed loop on system we use RPM counter sensor value as a feedback. Closed loop is most efficient approach as compared to open loop system. To implement this concept we derive some formulae and use them into our code.

Error = Reference value – Sensor value

We use check if error is greater than zero then it means that motor is running at lower speed as compared to reference value. For this case we add some duty cycle in pervious duty cycle value. We calculate that added value from above given formula, which is

Duty Cycle to compensate Error = 0.00625 x Error.

New Duty Cycle = Previous Duty cycle + Duty Cycle to compensate Error

If error is less than zero it means that motor is running at higher value as compared to reference value. So for this case formula will be,

Duty Cycle to compensate Error = 0.00625 x Error.

New Duty Cycle = Previous Duty cycle - Duty Cycle to compensate Error

We implement this concept and got this sort of result as shown in figure-2.4.4.2.

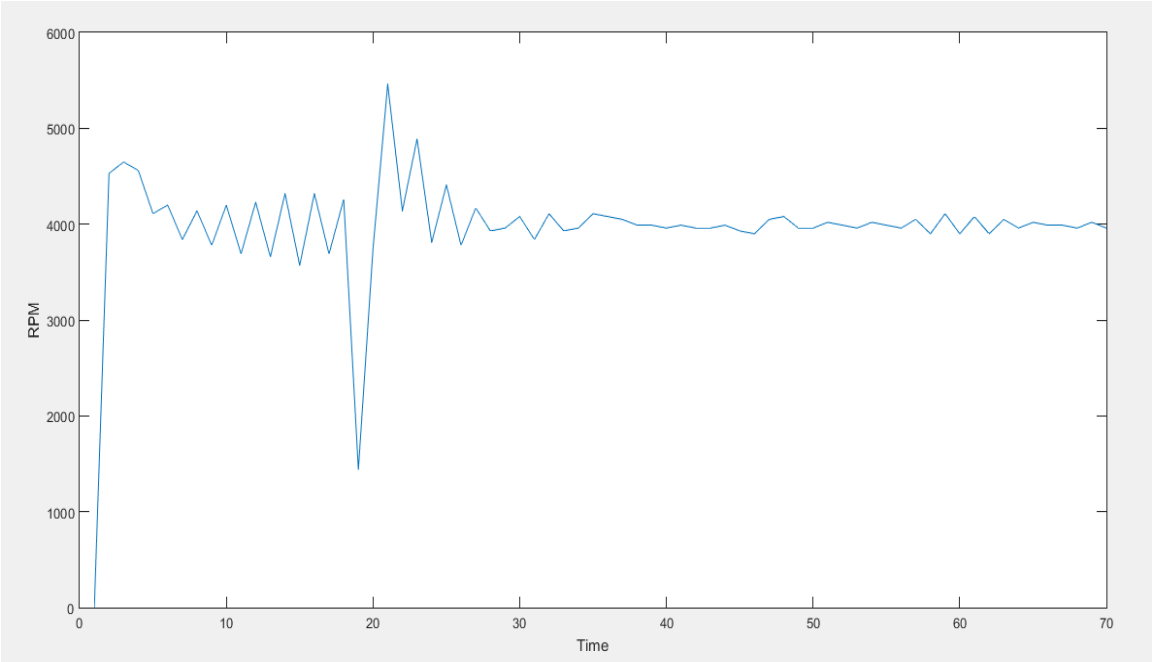


Figure-2.4.4.2: System behavior using Closed loop system at reference RPM = 4000

2.4.4.3 PID Implementation

For PID implementation we use the basic formula in our code, which is

Error = Reference value – Sensor value

Proportional = Kp x Error

Error Sum = Last Error + Error

Integral = Ki x Error Sum

Differential = Kd x (Error – Last Error)

After implementation the main thing was to tune the values of Kp, Ki and Kd. For this purpose we implement different techniques and chose which gives best result.

- **Zeigler Nicholas Method:**

We applied Zeigler Nicholas method for our system. We follow these steps and got pretty good results,

1- We run the motor at full rated RPM value (11800 RPM) by applying 100% duty cycle, and we got this plot on MATLAB.

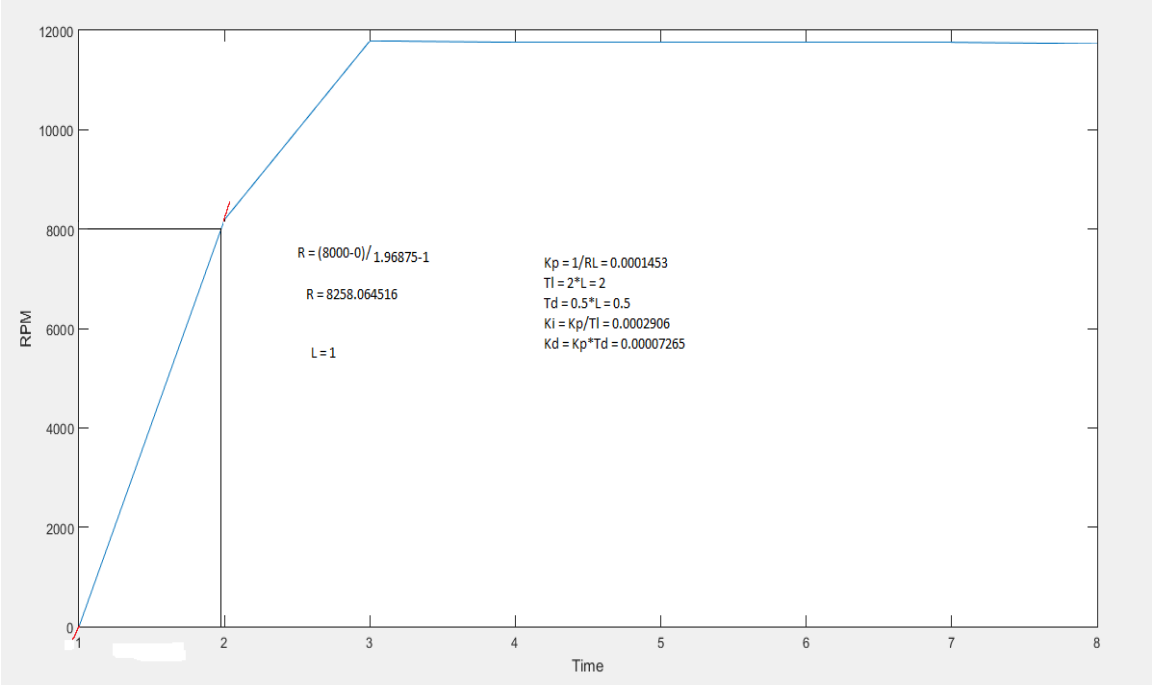


Figure-2.4.4.3: System behavior at full rated RPM

2- We calculate the values of L and R as it is written in graph and got
   Kp = 0.0001453
   Kd = 0.0002906
   Ki = 0.00007265
3- Put these value of into the code and set reference value of 3500 RPM and got these results on MATLAB

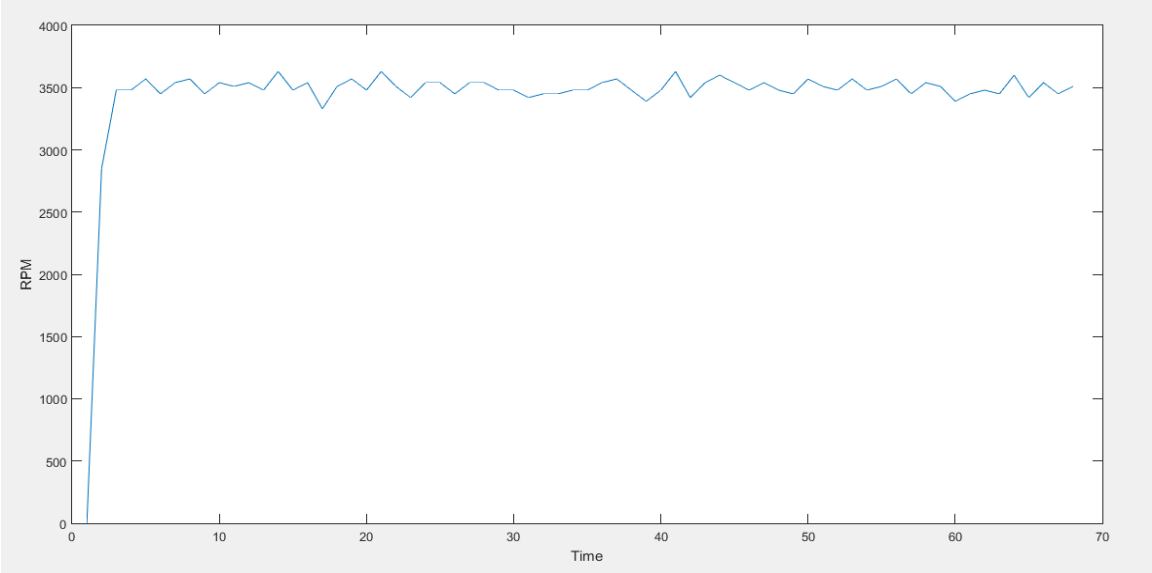


Figure-2.4.4.4: System behavior after PID tuning through Zeigler Nicholas Method

As we can see from the graph is that, there is still steady state error.

- **Ultimate Sensitivity Method:**

To implement this method follow these steps,

1- First of all we set Kp = Ki = Kd = 0.

2- Set a reference value (RPM) at which motor should run. Here reference RPM = 4500.

3- Then we start increasing the value of Ku until it start oscillating across the reference value as shown in graph below, figure 11. We get this response at Ku = 0.00011.

4- Then we calculate the value of time period Pu of oscillating wave. Pu is time period of marginally stable response. According to this graph time period Pu = 4 seconds.

Ku = 0.00011

Kp = 1.6 x Ku = 1.6 x 0.00011 = 0.000176

T1 = 0.5 x Pu = 0.5 x 4 = 2

Td = 0.125 x Pu = 0.125 x 4 = 0.5

Ki = Kp/T1 =0.000176/2 = 0.000088

Kd = kp x Td = 0.000176 x 0.5 = 0.000352

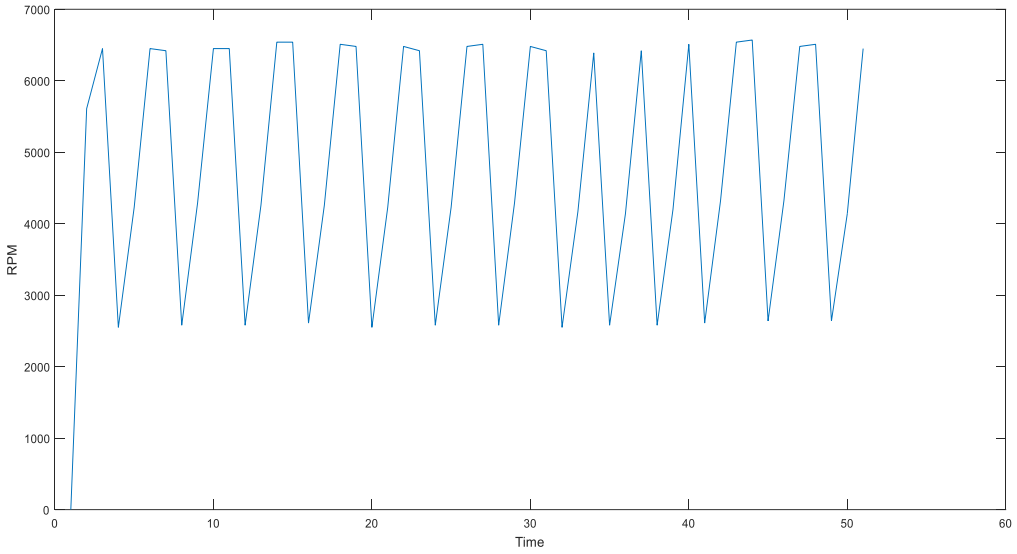


Figure-2.4.4.5: System behavior at Ku = 0.00011 which is Periodic

We put above calculated value of Kp = 0.000176, Ki = 0.000088 and Kd = 0.000352 into PID code and found this sort of response as shown in Figure-2.4.4.5.

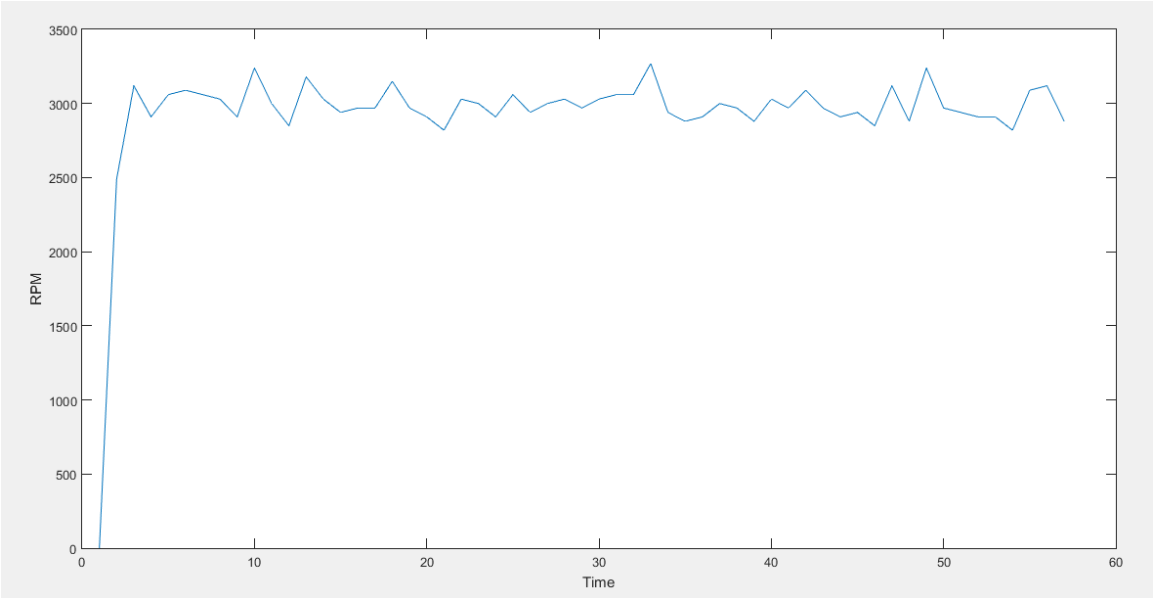


Figure-2.4.4.5: System behavior at reference RPM = 3000 using Ultimate Sensitivity Method

- **PID Manual Tuning:**

For PID tuning we follow these steps:

1- Set values of Kp, Ki and Kd to zero. Then start increasing the value of Kp from zero until it starts oscillating near reference value with reasonable overshoot.
2- Increase the value of Kd until the oscillations in steady state removed. But there was heavy overshoot.
3- We increase the value of Ki until the steady state error removed.

We got these values of Kp, Ki and Kd

Kp = 0.0001398
Kd = 0.0003146
Ki = 0.00007381

We got this graph,

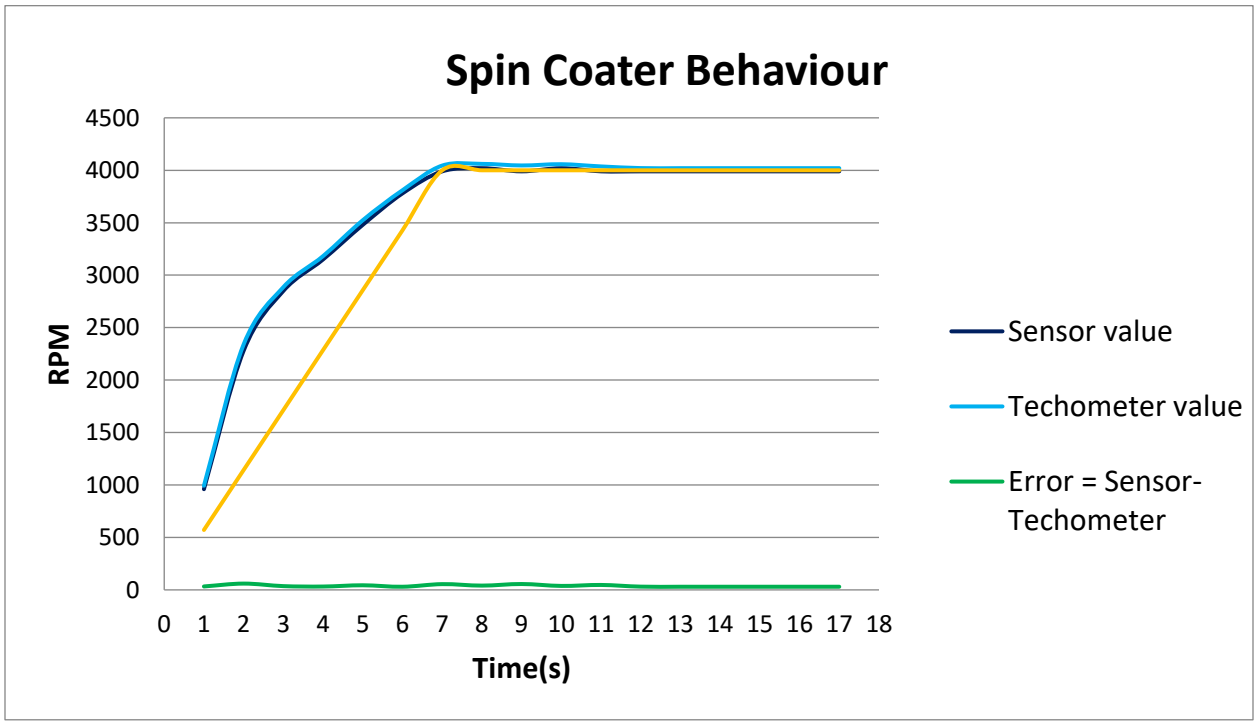


Figure-2.4.4.6: System behavior after Implementing Manual tunning at Reference RPM = 400

From the above graph we can see that the external tachometer value and systems sensor value are very near to each other, there is very small error. It means that system sensor is efficient and accurate to use.

- **Comparison of All techniques:**

All above techniques are very close to each other as compared to results but manual tuning is best technique, but manual tuning takes lot of time.

- **PID parameters and their effect:**

After tuning we observe that by increasing the value of any one variable (Kp, Ki, Kd) and keeping other two at constant value, we got this table

Name	**Rise Time**	**Over shoot**	**Settling time**	**Steady state error**
Kp	Reduced	Increased	Small change	Reduced
Kd	Reduced	Increased	Increased	Eliminate
Ki	Small change	Reduced	Reduced	Small change

Table 2.3.4 : PID parameters and their effects

2.5 Software

1) Arm MBED Compiler:

This software (fig-2.3.5.1) is used to compile code for mbed controllers. It is an online software which is lightweight cloud based and very suitable for IoT based technologies.

Fig.2.3.5.1

2) Proteus:

Figure (2.3.5.2) is logo of software called Proteus. Proteus is capable of providing platform to design a wide range of electric circuits and run simulations. Furthermore, by using proteus we make PCB design to make a circuit which is very small in size and error free.

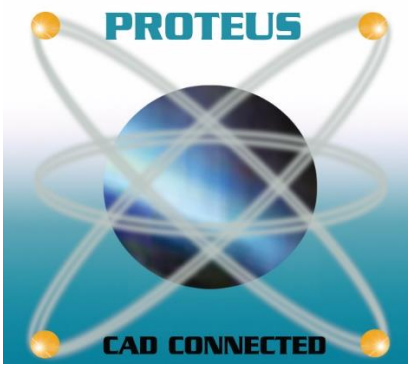


Fig- 2.3.5.2

3) MATLAB:

MATLAB is interactive environment for engineers and scientists which provides computing environment. In our project we used MATLAB to design PID and figure out the values of Kp , Kd and Ki for the desired output. The figure- below Fig- 2.3.5.3 shows logo of MATLAB.

Fig- 2.3.5.3

Chapter 3 (Spin Coating Procedures and Results)

3.1 Spin Coating Procedure

The spin coating technique of antenna comprises of five main steps. A series process is considered for uniform coating result free from Impurities. The following are the steps.

1. Cleaning the Substrate
2. Ultra Violet (UV) treatment
3. Masking the substrate
4. Spin coating
5. Curing/annealing

3.1.1 Cleaning the Substrate

Cleaning the substrate is essential to remove dust and other material from the surface of the substrate. Presence of impurity in the process damages the performance of the product.

Cleaning procedure includes following four steps.

1. Wash the substrate with **Distilled water** twice and let the substrate to dry.
2. Wash the substrate again by **Acetone** to remove any reactive particles.
3. Wash the substrate now by **Ethanol** to remove the residues of Acetone.
4. Wash the substrate finally again by the **Distilled water** to remove any residues of Ethanol and allow it to dry.

Cleaning procedure requires great attention and care. It should be done in a clean environment and by wearing the laboratory gloves as shown in Figure-3.1.1.

Figure-3.1.1 Proper handling of a substrate

3.1.2 Ultra Violet (UV) treatment

After cleaning, the substrate is subject to UV treatment for forty minutes. This is the key step to make the substrate **hydrophilic.** Hydrophilic means ‘water loving’ or polar solvents loving, means the capacity of the substrate to interact with the solvent molecules.

The time for UV treatment depends on the thickness and nature of the substrate. For 0.5 mm pure glass forty minutes are enough to make it hydrophilic. The UV treatment procedure is shown in Figure-3.1.2.

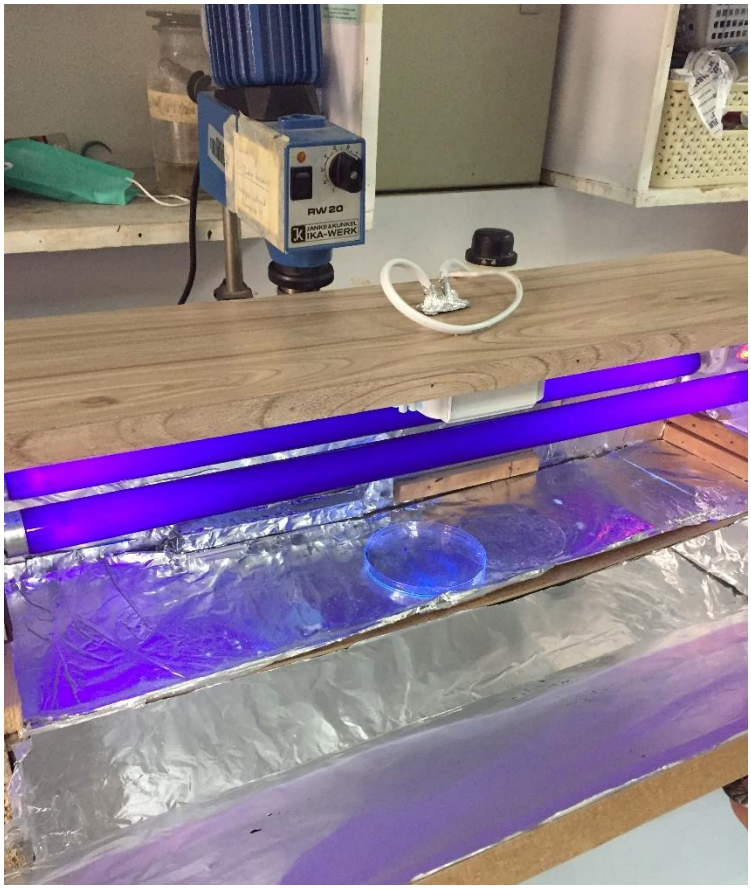


Figure-3.1.2 UV Treatment

3.1.3 Masking of Substrate

Masking is the step where the cleaned and UV treated substrate is shaped into the required design. This is done using a sticker paper to cover the parts of substrate where coating of material is not required. Rest of the substrate is exposed for the coating. The Masking procedure is shown in figure-3.1.3.

Figure-3.1.3 Masking of substrate

3.1.4 Ink drop and Spin Coating

After cleaning, UV treating and masking the substrate is now ready for spin coating. Coating procedure includes the following steps as under:-

1. Mount the masked substrate on the spin coater rotating plate with help of screw nuts or double sided stick tape.
2. Set the desire/require RPM and time using the input keypad of Spin Coater.
3. Take the coating polar solvent in a syringe and be ready.
4. Start the Spin Coater and allow it to reach the required speed.
5. Drop the ink on the center of the substrate in the form of drops.
6. Repeat the process if more thickness is required.
7. Remove the substrate carefully.

The spin coating step-up is shown in figure-3.1.4.

Figure-3.1.4 Spin coater with masked substrate

3.1.5 Curing

The curing process requires a laboratory heating oven. After spin coating the substrate is put into the oven and heat it up to 120 degree Celsius for one hour. Curing process makes the coating surface uniform and in result properties such as radiation parameters improve, in case of antenna.

Upon completing the curing process take out the substrate from the oven and allow it to cool. Now remove the masking tape and the antenna is ready.

The curing process is shown in figure-3.1.5.

Figure-3.1.5 A electric Oven operating at 204 C

3.2 Spin Coating Results

3.2.1 Results using Profilo-metery

The 2D Non Contact Profilometer model PS-50, NANOVEA ,USA was used to find out the results of spin coating. The Sample Name is 'Titanium Oxide Thinfilm on Glass Slide' which was provided for tests.

The following results were obtained for various properties of the coat on the glass.

3.2.1.1 Height Profile

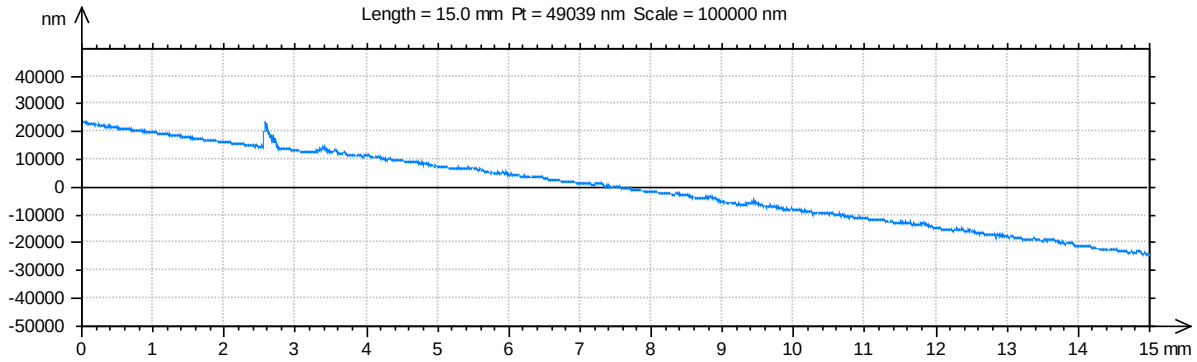


Figure-3.2.1.1 Graph of thickness a specific area

3.2.1.2 Intensity Profile

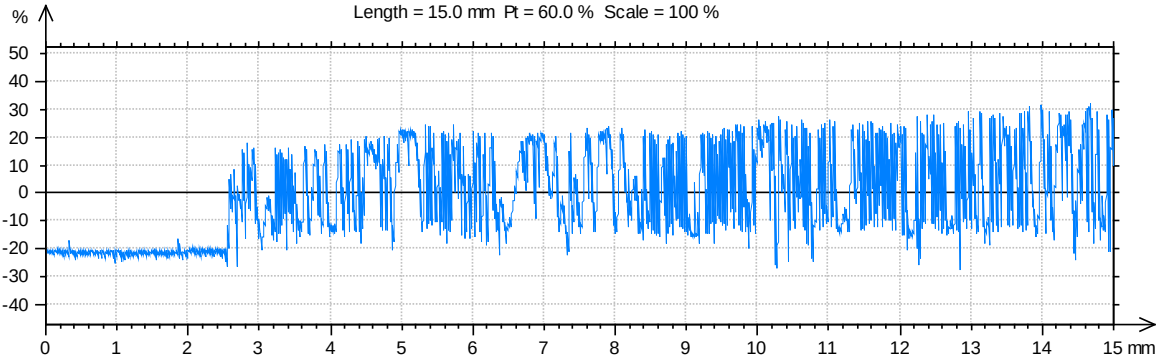
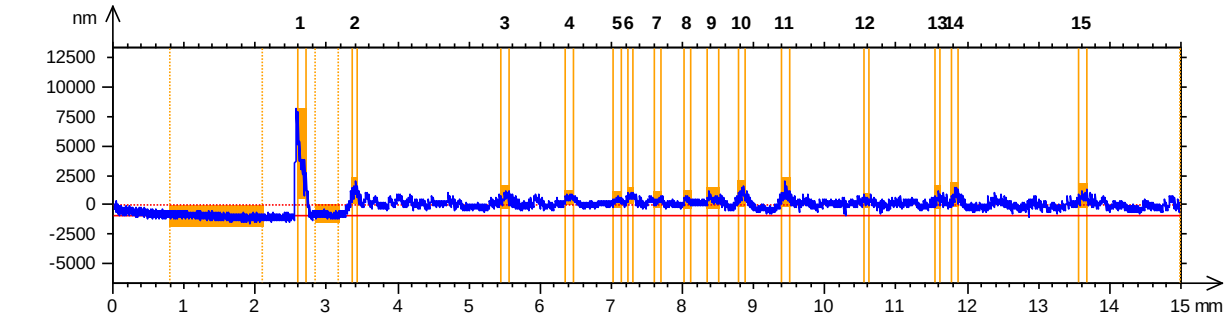


Figure-3.2.1.2

3.2.1.3 Thickness Profile



Mean values on 15 steps.

Maximum height	8903 nm
Mean height	1781 nm
Width	0.0997 mm

Figure-3.2.1.3 Graph of height vs length

3.2.1.4 Roughness Profile

ISO 4287			
Amplitude parameters - Roughness profile			
Rp	1006	nm	*Gaussian filter, 0.8 mm*
Rv	694	nm	*Gaussian filter, 0.8 mm*
Rz	1700	nm	*Gaussian filter, 0.8 mm*
Rc	524	nm	*Gaussian filter, 0.8 mm*
Rt	2903	nm	*Gaussian filter, 0.8 mm*
Ra	218	nm	*Gaussian filter, 0.8 mm*
Rq	276	nm	*Gaussian filter, 0.8 mm*
Rsk	0.486		*Gaussian filter, 0.8 mm*
Rku	3.54		*Gaussian filter, 0.8 mm*
Material Ratio parameters - Roughness profile			
Rmr	0.914	%	*c = 1000 nm under the highest peak, Gaussian filter, 0.8 mm*
Rdc	446	nm	*p = 20%, q = 80%, Gaussian filter, 0.8 mm*

Figure-3.2.1.4 Table shows uniformity and roughness of coating

3.3 Spin coated WLAN Antenna

3.3.1 Introduction

In today's world Wireless Local Area Network (WLAN) is one of the commonly used technology. It is popular because it is a reliable, cost effective and fast as it provides high speed data connectivity. The standard Wi-Fi protocols of IEEE 802.11b and 802.11g use 2.45 GHz, whereas IEEE 802.11ah uses 900 MHz. It is requirement of the time to integrate various WLAN technologies band into a single unit [7]. The fast evolving technologies demand robust, compact and highly efficient antennas in order to fulfill the desired operations. A number of single and dual band antennas are reported to cover various WLAN bands such as 2.4 GHz and 5.2 GHz etc [8].

On the other hand, flexible technologies have made a rapid progress in world of electronics and we have witnessed rollable keyboards, flexible displays, and flexible smart sensors [10].

The above mentioned technologies demand antenna system integration in order to get wireless connectivity and the antenna should need to be ultra-light, thin and flexible along with that it should have characteristics such as robustness and efficiency to produce desirable radiation.

The flexibility of the antenna depends on the substrate and it needs to be established before going for the spin coating or printing process that whether the substrate and the conductive ink has any adherence. The coating will only be possible if there is adherence.

3.3.2 Finite element method

Partial differential equations (PDEs) are usually used to express the laws of space and time dependent problems in physics. When it comes to solve majority of problems with analytical methods, PDEs are useless for a wide range of geometries and problems. Alternatively, based on types of discretization an equation is constructed that approximates the equations. Using numerical methods, the PDEs are approximated with numerical model equations. The solution to these equations provide approximation of the real solution to the PDEs. These approximations are computed through finite element method (FEM).

3.3.3 Antenna Design

Printed antenna is designed by finite element method (FEM) on software such as Ansys high frequency structure simulator (HFSS) and others. Various shape and sizes antennas are designed for single band and dual band functionalities. Just to give basic introduction about these antennas and their geometry we discuss single band and dual band antennas below. We chose dual band WLAN 900 MHz and 2.4 GHz antenna designed by [8].

3.3.3.1 Single band printed Antenna

The 2.4 GHz is a single band U-shaped monopole. The shape of the antenna is desired to be kept in a way that the structure size reduces without significant degradation in efficiency. In the case of the below monopole antenna picked from source [9] as shown in figure-3.3.3.1 ,the two arms of antenna are 6 mm apart which achieves the least return loss. A smaller separation between the arms leads to increase in capacitive coupling which results in impedance mismatch. The U-shape monopole is fed by a microstrip line which is 1.5mm wide 50 ohm. The size of the antenna is 26.5 mm x 25 mm and both monopole and microstrip line are printed on same side, this is the front side of the substrate. On the back side of the substrate a copper ground plane is adhered which is 12.5 mm x 25 mm in size and below the microstrip line. The length of the U-shaped monopole controls the resonance frequency as well [9].

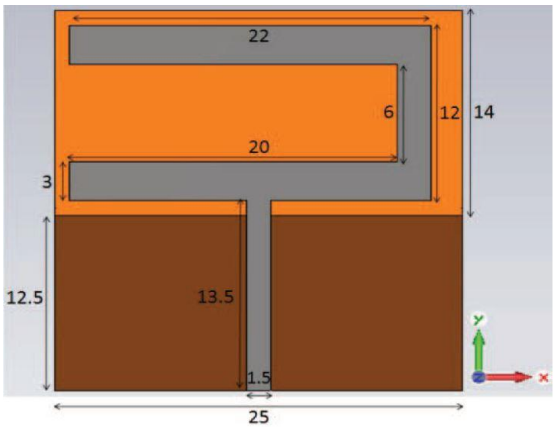
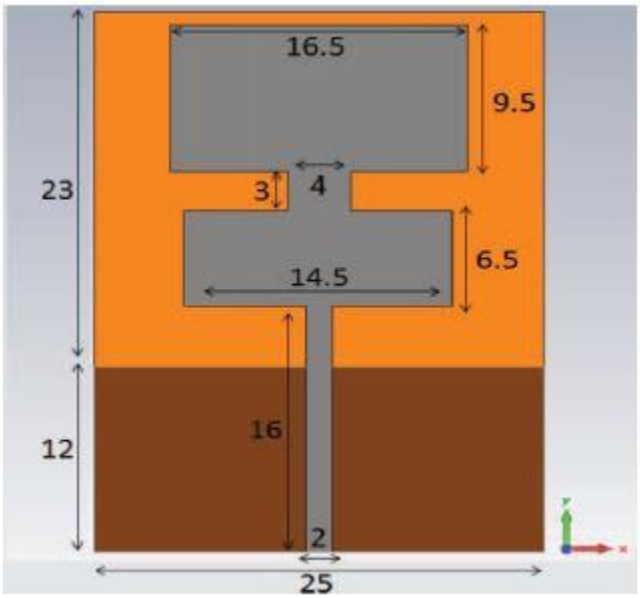


Figure-3.3.3.1 Monopole Antenna

3.3.3.2 The dual band antenna:

Here we will shed light over two models which are presented by different researchers in [8] and [9]. Firstly, the proposed model in [9] for dual band antenna consists of two monopole T-shaped branches operating over 2.5 GHz and 5.2 GHz. A 2 mm wide 50 ohm microstrip line is fed to these two monopoles. The recommended substrate for this design is Kapton Polyimide which is a flexible material. On front side of the substrate the two branches and microstrip line is printed in 35mm x 25 mm size. On the back side a copper ground plane is adhered below the microstrip in 12 mm x 25 mm size area. The main controlling factor in the design are the widths and lengths of the branches as they control resonance frequency of the antenna. In the dual band antenna the upper monopole branch controls the 2.5 GHz mode and the lower branch controls the 5.2 GHz band. The controlling parameters were adjusted to accomplish a dual resonance at 2.5 GHz and 5.2 GHz which is appropriate for the proposed WLAN applications. It is worth noting here that the ground plane can affect the bandwidth and resonant frequency of both modes [9]. Below figures 3.3.3.2 show the proposed dual band antenna model presented in source [9].



Figures 3.3.3.2

3.3.4 The spin coated Antenna

Now, we come towards the 2nd model which is presented in source [9] and we put our efforts into remaking this model in practical by using our own manufactured Spin coater through the process of spin coating.

The original design of this antenna is picked from [9]. Where the substrate used was flexible polyethylene terephthalate (PET) and the conductive ink of silver nanoparticles (AgNPs) is used for coating. In our case, the substrate is 2 mm thick glass slide and gold ink was used due to unavailability of silver conductive ink. Once the spin coating process was carried out by masking the glass substrate, the coated circuit was connected with a female Subminiature version A (SMA) connector by using silver epoxy paste to make the connection between the coated conductive gold surface and SMA connector pins.

The dual band antenna is designed for 900 MHz and 2.4 GHz. The antenna structure in [9] is a Z-shape radiating monopole served by co-planar waveguide (CPW). CPW provides key advantage by allowing both radiating body and ground plane to be printed on the same side of substrate which reduces production cost [9].

This antenna design has 3 arms, the acquire optimum separation between the arms is 9 mm with 50 ohm input. Furthermore, if the separation is smaller than 9mm then the coupling effect between arms increases which worsens impedance matching [9]. The monopole antenna has ground as radiating surface which reduces the current leakage and electrical surge in the antenna circuit. A method is adapted to achieve the dual band characteristics in this antenna; it is done by making rectangular slots inside the plane and in result making 3 arms of the antenna. The length of the arm is adjusted in a way that it would radiate at specific band and provide the desired band width and gain.

The table shown below table-3.3.4 and figure 3.3.4a contains the design parameters and shape of the antenna.

Parameter	Dimension (nm)	Parameter	Dimension(mm)
L	45	D	11.86
W	36	Ws	9
Lg	30	Ls	27
Wg	25.02	Lf	41.86
Wt	9	Wf	3.5

Table 3.3.4 shows the measurement of the Dual band antenna

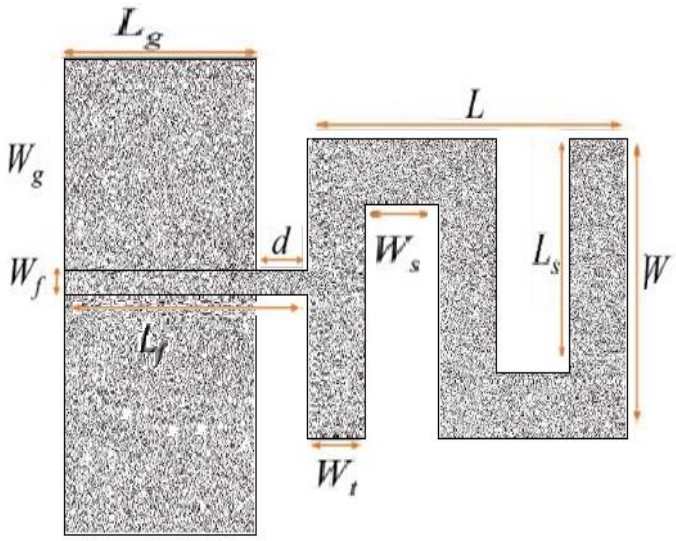


Figure 3.3.4a

The figure 3.3.4b shown on the below is the gold spin coated antenna on glass substrate with SMA connector. The antenna is fabricated using the spin coater we made as the project. The surface of antenna is conductive and it is verified by using digital multi-meter.

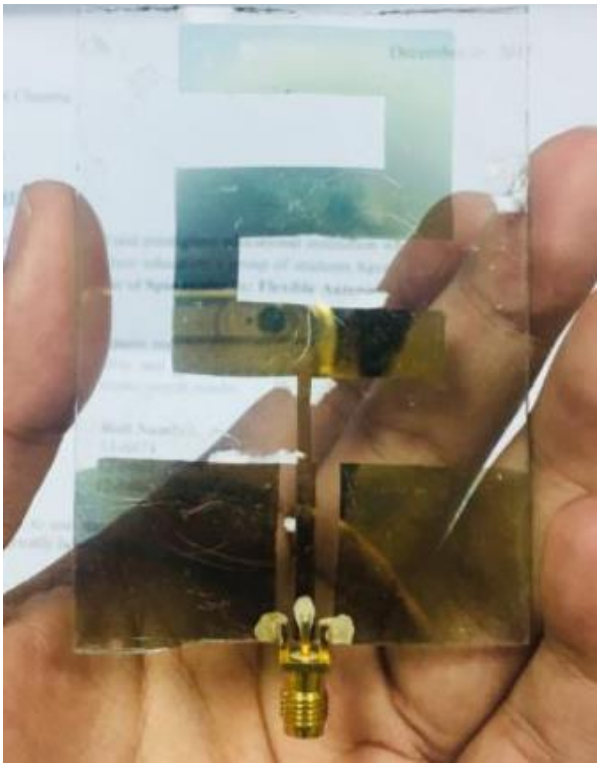
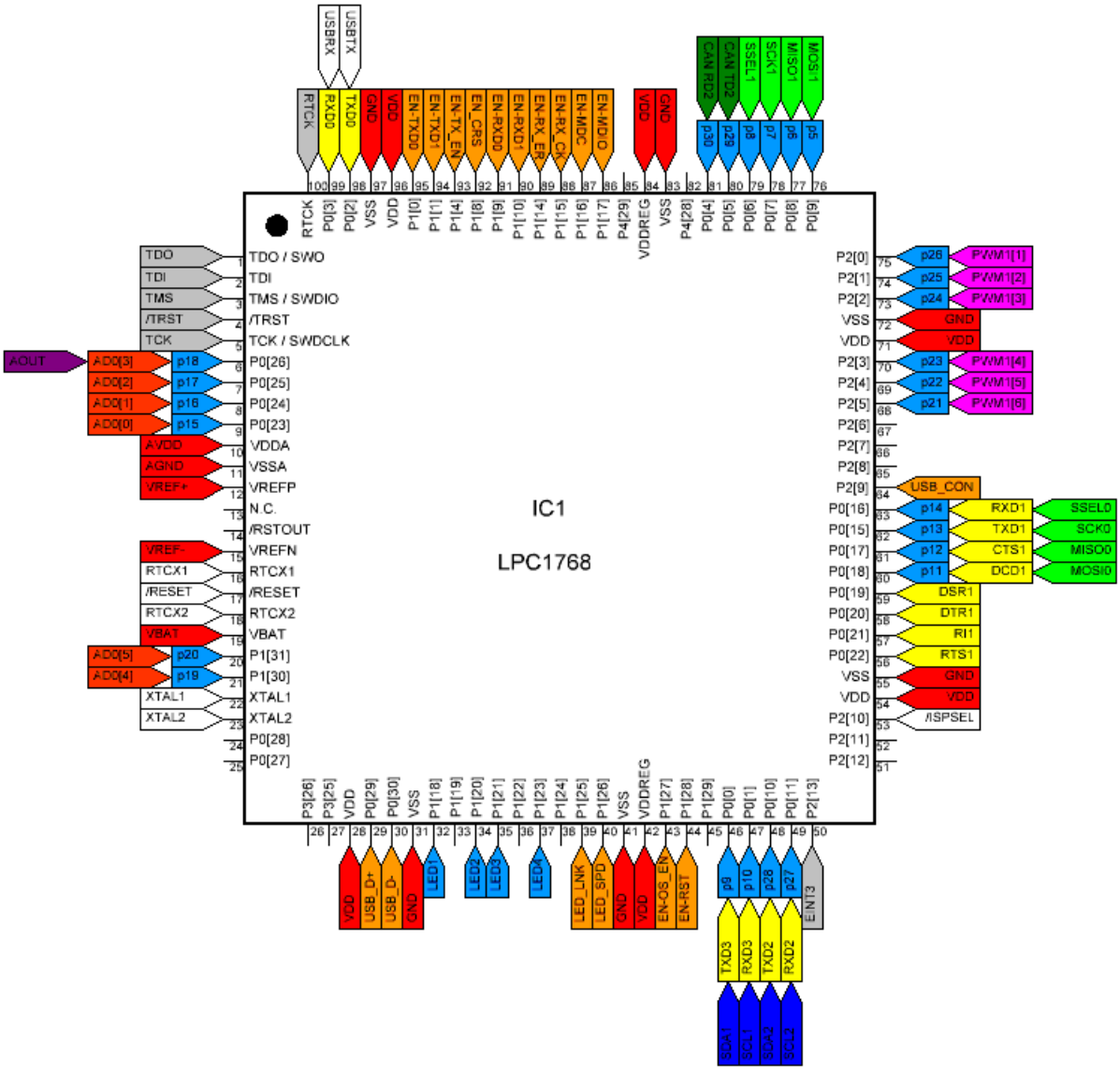


Figure-3.3.4b

Appendix A

Pin out Diagram of microcontroller LPC-1768.



Appendix B

Code:

```
#include "mbed.h"

#include "Servo.h"

Serial pc(USBTX, USBRX);

BusOut
data(p22,p23,p24,p25,p26,p27,
p28,p29);

DigitalOut r1(p5);

DigitalOut r2(p6);

DigitalOut r3(p7);

DigitalOut r4(p8);

DigitalIn c1(p9);

DigitalIn c2(p10);

DigitalIn c3(p11);

DigitalIn c4(p12);

//////////////////PWM and
sensor veriables

Servo myservo(p21);

InterruptIn sensor(p13);

int rpm_counter=0;

int Sensor_value;
//////////Sensor value

PwmOut ledf(LED1);

PwmOut ledr(LED2);

DigitalOut ss(p18); /////red
lower

DigitalOut start(p19);
///blue upper

DigitalOut ramp(p20);
/////center green

//////////////////

DigitalOut rs(p30);

DigitalOut en(p14);

  float Ramp_Fraction;

 int
dcount11,dcount12,dcount13,dc
ount14; ////////// for digit
count

 int sdigits[10];

 int size;

 float factor = 0.05; //////
start duty cycle

 int new_reference = 0;

 int j=0;

 int d;  //////////d----
>digit

 int p; //////////for
conversion process

 int
Ramp_RPM,Ramp_Time,SS_RPM,SS_
Time;

 int Save_RPM;

 int Error;////////////Error
in feedbackloop

 int RampRPM_Array[3];

 int RampTime_Array[3];

 int SSRPM_Array[3];

 int SSTime_Array[3];

 float Matlabdata; //////

 ////////////
```

```
 int Error_Sum;
 float Kp = 0.0001398;
 float Ki = 0.00007381;
 float Kd = 0.0003146;
 float P;
 float I;
 float D;
 float PID;
 int Last_Error;
 int Divider = 96000/12000;
/////value of divider to
scale down pid value = Max
pid value/Max rpm at 11.5
volt
 ////////////
 Ticker timeup;
 Ticker over;
 void comandwrt()
 {
     rs=0;
     en=1;
     wait(0.0023);
     en=0;
     }
     void datawrt()
     {
         rs=1;
         en=1;
         wait(0.0023);
         en=0;
         }
```

```
//////////////////////////
/////////////////

//////////////////////////
/////////////////
          int keypadN()
          {

           int value;
           if(c1==0)
           {
                r1=1; r2=0;
r3=0; r4=0;

                //wait(0.001);
                if(c1==1)
                {

value=49;
                          ///1
is pressed
                }
                //////////////
                r1=0; r2=1;
r3=0; r4=0;

                //wait(0.001);
                if(c1==1)
                {

value=52;
                          ///4
is pressed
```

```
                }

///////////////r=3/
                r1=0; r2=0;
r3=1; r4=0;

                //wait(0.001);
                if(c1==1)
                {

value=55;
                        ///7
is pressed
                }

///////////////r=4
                r1=0; r2=0;
r3=0; r4=1;


///wait(0.0000001);
                if(c1==1)
                {

value=49;
                        ///*
is pressed
                }

            }


//////////////////////////////
/////////// for c2
            else if(c2==0)
          {
                r1=1; r2=0;
r3=0; r4=0;

                //wait(0.001);
                if(c2==1)
                {

value=50;
                            ///2is
pressed
                }
                //////////////
                r1=0; r2=1;
r3=0; r4=0;

                //wait(0.001);
                 if(c2==1)
                {

value=53;
                            ///5 s
pressed
                }

///////////////r=3/
                r1=0; r2=0;
r3=1; r4=0;
```

```
              //wait(0.001);
              if(c2==1)
              {
value=56;
                        ///8 s
pressed
              }
///////////////r=4
              r1=0; r2=0;
r3=0; r4=1;

              //wait(0.001);
              if(c2==1)
              {
value=50;
                        ///0
is pressed
              }

            }
//////////////////////////
///////////for c=3
            else if(c3==0)
          {
              r1=1; r2=0;
r3=0; r4=0;

              //wait(0.001);
              if(c3==1)
              {
value=51;
                        ///3
is pressed
              }
//////////////
              r1=0; r2=1;
r3=0; r4=0;

              //wait(0.001);
              if(c3==1)
              {
value=54;
                        ///6 s
pressed
              }
///////////////r=3/
              r1=0; r2=0;
r3=1; r4=0;

              //wait(0.001);
              if(c3==1)
              {
value=57;
                        ///9 s
pressed
              }
///////////////r=4
```

```
                r1=0; r2=0;
r3=0; r4=1;


                //wait(0.001);
                if(c3==1)
                {

value=51;

///Actually key in keypad is
# but i assign ascii of A
                }

            }


////////////////////
//////////////////////////////
////////////c4
         else if(c4==0)
          {
                r1=1; r2=0;
r3=0; r4=0;


                //wait(0.001);
                if(c4==1)
                {

value=65;
                        ///A
is pressed
                }
                //////////////
                r1=0; r2=1;
r3=0; r4=0;


                //wait(0.001);
                if(c4==1)
                {

value=48;
                            ///B s
pressed
                }

///////////////r=3/
                r1=0; r2=0;
r3=1; r4=0;


                //wait(0.001);
                if(c4==1)
                {

value=48;
                            ///0
is pressed
                }

///////////////r=4
                r1=0; r2=0;
r3=0; r4=1;


                //wait(0.001);
                if(c4==1)
```

```
              {

value=65;

///Actually key in keypad is
* but i assign ascii of A

              }


            }


//////////////////////////////
///////////////

              return value;


     }


//////////////////////////////
///////

     int Converter(int
dcount)
/////////////////Ascii to
decimal conversion

     {


         if(data==48)

         d=0;

         if(data==49)

         d=1;

         if(data==50)

         d=2;

         if(data==51)

         d=3;

         if(data==52)

         d=4;

         if(data==53)

         d=5;

         if(data==54)

         d=6;

         if(data==55)

         d=7;

         if(data==56)

         d=8;

         if(data==57)

         d=9;


         if(dcount==0)

          p=d;
////////e.g  9

         if(dcount==1)

         {

            p=p*10;
////////for second digit
p=9x10=90,  90+2=92   here i
pick 2 as example

            p=p+d;

          }

         if(dcount==2)

         {

            p=p*10;
///////////so on for others

            p=p+d;

          }


         if(dcount==3)
```

```
        {
           p=p*10;
           p=p+d;
         }
         return p;
//////////////return decimal
digit
    }////////end of function
    ////////////////////////
    void SSTime_Display()
    {

              data='S';
              datawrt();
              data='S';
              datawrt();
              data=' ';
              datawrt();
              data='T';
              datawrt();
              data='i';
              datawrt();
              data='m';
              datawrt();
              data='e';
              datawrt();
              data=' ';
              datawrt();
              data=':';
              datawrt();
              data=' ';
              datawrt();

    }
    ////////////////////
    void SSTime()
    {       p=0;
            d=0;
            dcount14=0;
            data=0X94;
            comandwrt();

SSTime_Display();
            data=0;
            while(data!=65)
            {
                if(c1==0 |
c2==0 | c3==0 | c4==0)
                {

data=keypadN();

if(data==65)
                    {

wait(0.5);

break;
                    }
                    else
                    {

datawrt();
```

```
wait(0.5);

SSTime_Array[dcount14]=data;

SS_Time=Converter(dcount14);

dcount14++;
                              }
                        }////end
of if condition

             }/////end of
while
      }

//////////////////////////
      void SSRPM_Display()
      {
             data=0XC0;
             comandwrt();

             data='S';
             datawrt();
             data='S';
             datawrt();
             data=' ';
             datawrt();
             data='R';
             datawrt();
             data='P';
             datawrt();
             data='M';
             datawrt();
             data=' ';
             datawrt();
             data=':';
             datawrt();
             data=' ';
             datawrt();
       }
//////////////////////
      void SSData()
       {
             data=' ';
             datawrt();
             data=' ';
             datawrt();
             data=' ';
             datawrt();
             data=' ';
             datawrt();
             data=' ';
             datawrt();
             data='S';
             datawrt();
             data='S';
             datawrt();

             datawrt();
             data=' ';
             datawrt();
             data='D';
```

```
        datawrt();
        data='a';
        datawrt();
        data='t';
        datawrt();
        data='a';
        datawrt();
       }
void SSRPM()
{
        p=0;
        d=0;
        dcount13=0;

        data=0X01;
        comandwrt();
        data=0X0E;
        comandwrt();
        data=0X06;
        comandwrt();
        data=0X38;
        comandwrt();
        data=0X80;
        comandwrt();
        SSData();
        SSRPM_Display();

        data=0;
        while(data!=65)
        {
                    if(c1==0 |
c2==0 | c3==0 | c4==0)
                    {

data=keypadN();

if(data!=65)
                        {

datawrt();

wait(0.5);

SSRPM_Array[dcount13]=data;
//////////to save ascii of a
digit in array

SS_RPM=Converter(dcount13);
///////ascii to decimal
conversion

dcount13++;
/////////////////digit counter
                            }
                         else
                        {

wait(0.5);

SSTime();
                            }

                       }////end of
if condition
```

```
            }/////end of
while
      }

////////////////////////////
//
      void RampTime_Display()
      {
            data=0X94;
//////must be on third line
            comandwrt();


            data='R';
            datawrt();
            data='a';
            datawrt();
            data='m';
            datawrt();
            data='p';
            datawrt();
            data=' ';
            datawrt();
            data='T';
            datawrt();
            data='i';
            datawrt();
            data='m';
            datawrt();
            data='e';
            datawrt();
            data=' ';
            datawrt();
            data=':';
            datawrt();
            data=' ';
            datawrt();


            }

////////////////////////////
//////////////////
     void RampTime()
     {
            p=0;
            d=0;
            dcount12=0;

RampTime_Display();


            data=0;
            while(data!=65)
            {
                if(c1==0 |
c2==0 | c3==0 | c4==0)
                {

data=keypadN();

if(data!=65)
                    {

datawrt();

wait(0.5);
```

```
RampTime_Array[dcount12]=data
;

Ramp_Time=Converter(dcount12)
;

dcount12++;

                        }
                       else
                      {

wait(0.5);

SSRPM();  /////////after
returning from this function
system will check 65(while
given above)

                          }
                      }////end
of if condition

             }/////end of
while
      }

////////////////////////////
//////////////////
      void
Rampdata()/////Ramp data
heading
      {
             data=' ';
             datawrt();
             data=' ';
             datawrt();
             data=' ';
             datawrt();
             data='R';
             datawrt();
             data='A';
             datawrt();
             data='M';
             datawrt();
             data='P';
             datawrt();
             data=' ';
             datawrt();
             data='D';
             datawrt();
             data='a';
             datawrt();
             data='t';
             datawrt();
             data='a';
             datawrt();


 }
 void RampRPM_Display()
{
             data=0XC0;
             comandwrt();
             data='R';
             datawrt();
             data='a';
```

```
            datawrt();
            data='m';
            datawrt();
            data='p';
            datawrt();
            data=' ';

            data='R';
            datawrt();
            data='P';
            datawrt();
            data='M';
            datawrt();
            data=' ';
            datawrt();
            data=':';
            datawrt();
            data=' ';
            datawrt();
         }

//////////////////////////////
//////////////////

     void RampRPM()
     {
            d=0; /////no
digit at start
            p=0; ////// for
conversion process
            dcount11=0;
            data=0X01;
            comandwrt();
            data=0X0E;
            comandwrt();
            data=0X06;
            comandwrt();
            data=0X38;
            comandwrt();
            data=0X80;
            comandwrt();
            Rampdata();

RampRPM_Display();
            data=0;

            while(data !=65)
            {
                if(c1==0 |
c2==0 | c3==0 | c4==0)
                {

data=keypadN();

if(data!=65)
                    {

datawrt();

wait(0.4);

RampRPM_Array[dcount11]=data;
```

```
//////////////To save ascii
RampRPM ascii

Ramp_RPM=Converter(dcount11);
/////////Ascii to decimal
digit converter function



dcount11=dcount11+1;
                        }
                         else
                        {

wait(0.5);


RampTime();
                          }

                }////end of
if condition

                }/////end of
while

      }

//////////////////////////

/////////////////////////////
//

/////////////////////////////
///
      void spincoater()
      {
                        data=0X01;
                        comandwrt();
                        data=0X0E;
                        comandwrt();
                        data=0X06;
                        comandwrt();
                        data=0X38;
                        comandwrt();
                        data=0X86;
                        comandwrt();


                        data='W';
                        datawrt();
            //////////////Wellcome
                        data='E';
                        datawrt();
                        data='L';
                        datawrt();
                        data='L';
                        datawrt();
                        data='C';
                        datawrt();
                        data='O';
                        datawrt();
                        data='M';
                        datawrt();
                        data='E';
                        datawrt();
```

```
                data=0XC4;
//Next
                comandwrt();



                data='S';
                datawrt();
                data='P';
                datawrt();
                data='I';
                datawrt();
                data='N';
                datawrt();
                data=' ';
                datawrt();
                data='C';
                datawrt();
                data='O';
                datawrt();
                data='A';
                datawrt();
                data='T';
                datawrt();
                data='E';
                datawrt();
                data='R';
                datawrt();


                data=0XD6;
                comandwrt();
                data='A';
                datawrt();
                data='-';
                datawrt();
                data='-';
                datawrt();
                data='-';
                datawrt();
                data='-';
                datawrt();
                data='>';
                datawrt();
                data='F';
                datawrt();
                data='o';
                datawrt();
                data='r';
                datawrt();
                data=' ';
                datawrt();
                data='M';
                datawrt();
                data='e';
                datawrt();
                data='n';
                datawrt();
                data='u';
                datawrt();
```

```
          }
          int
DecimalToAscii(int x)
          {
               int y;
               if(x==0)
               y=48;
               if(x==1)
               y=49;
               if(x==2)
               y=50;
               if(x==3)
               y=51;
               if(x==4)
               y=52;
               if(x==5)
               y=53;
               if(x==6)
               y=54;
               if(x==7)
               y=55;
               if(x==8)
               y=56;
               if(x==9)
               y=57;

               return y;

}////////////end of function
```

```
////////////////////////////
//
               void
Ramp_Stage()
               {
                   data='R';
              datawrt();
              data='a';
              datawrt();
              data='m';
              datawrt();
              data='p';
              datawrt();
              data=' ';
              datawrt();
              data='S';
              datawrt();
              data='t';
              datawrt();
              data='a';
              datawrt();
              data='g';
              datawrt();
              data='e';
              datawrt();

          }

////////////////////////////
////////
```

```
          void SS_Stage()
              {
                  data='S';
             datawrt();
             data='S';
             datawrt();
             data=' ';
             datawrt();
             data='S';
             datawrt();
             data='t';
             datawrt();
             data='a';
             datawrt();
             data='g';
             datawrt();
             data='e';
             datawrt();


          }

////////////////////////////
//


////////////////////////////
//////////////////////

     void RunTimeDisplay()
//////////////////////IS
Function man masla ha

       {
        data = 0;
        }
```

```
////////////////////////////
/////////////////

////////////////////////////
/////////////////

////////////////////////////
//////////////////

////////////////////////////
////////////////////////////
////////

        void
RunTimeDisplay2()
//////////////////////IS
Function man masla ha

       {
             int x,y,i=0;
             int
Time1_Array[3];
             //float
Ramp_Fraction;

///Ramp_Fraction=Ramp_RPM/Ram
p_Time; //////////////RPM
increment veriable

             data=0X01;
             comandwrt();
             data=0X0E;
             comandwrt();
             data=0X06;
             comandwrt();
             data=0X38;
             comandwrt();
             data=0X85;
```

```
            comandwrt();
            SS_Stage();

            data=0XC0;
            comandwrt();
            SSRPM_Display();
            for(int t=0;
t<dcount13; t++)
            {

data=SSRPM_Array[t];
                  datawrt();
               }
            data=0;

while(SS_Time>=0)
            {

///PWM1=Ramp_Fraction/1000;
/////////////////Duty Cycle
            i=0;
            y=SS_Time;
            while(data!=65)
////////////// complex logic to
get out from this loop
              {
                  x=y%10;
                  y=y/10;
////Conversion from decimal
number to separte digits

Time1_Array[i]=DecimalToAscii
(x);
                  i++;
                  if(y<10)
                  {

Time1_Array[i]=DecimalToAscii
(y);

data=65;
                        }
               }
/////////////////////end of
while


            data=0X94;
            comandwrt();

SSTime_Display();


             for(int j=i;
j>-1; j--)
            {

data=Time1_Array[j];
////////Display of characters

datawrt();
            }


SS_Time=SS_Time-1;
            wait(1);
```

```
            }
///////////////Ramp_Time  end
loop

wait(1);////////////Faz
Khate man

data=65;//////// to get exit
path from while loop in main


        }///////End of
RunTimeDisplay

////////////////////////////
//////////////////////

////////////////////////////
//////////////////////

  void Process_Complete()
  {
            data=0X01;
            comandwrt();
            data=0X0E;
            comandwrt();
            data=0X06;
            comandwrt();
            data=0X38;
            comandwrt();
            data=0XC6;
            comandwrt();

            data='P';
            datawrt();
            data='r';
            datawrt();
            data='o';
            datawrt();
            data='c';
            datawrt();
            data='e';
            datawrt();
            data='s';
            datawrt();
            data='s';
            datawrt();

            data=0X9A;
            comandwrt();

            data='C';
            datawrt();
            data='o';
            datawrt();
            data='m';
            datawrt();
            data='p';
            datawrt();
            data='l';
            datawrt();
            data='e';
            datawrt();
            data='t';
            datawrt();
```

```
            data='e';
            datawrt();
            wait(3);
            data=65;

     } ////////////End of
process complete

////////////////////////////
///////

            ////////////ISR
           void
RampCalculations()
            {
                  int
local_rpm=0;
                  factor =
0.08;
                  myservo =
factor;
                  j = 0;

new_reference = new_reference
+ Ramp_Fraction; ////////////
                  while (j
== 0)
                      {

wait(0.3);

local_rpm = rpm_counter;

if(((local_rpm*60/(2*0.3)) -
(local_rpm*70)) <
new_reference)
{

factor = factor +
(0.001*Ramp_Fraction/90);

myservo = factor;

}

////////////////////

wait(0.3);

local_rpm = rpm_counter;

if(((local_rpm*60/(2*0.6)) -
(local_rpm*20)) <
new_reference)

{

factor = factor +
(0.001*Ramp_Fraction/90);

myservo = factor;

}

//////////////////////

wait(0.3);

local_rpm = rpm_counter;

if(((local_rpm*60/(2*0.9)) -
(local_rpm*3.333)) <
new_reference)
```

```
{

factor = factor +
(0.001*Ramp_Fraction/90);

myservo = factor;

}

wait(0.06);

                      }
//////end of while loop
          factor = factor -
0.02;

          myservo = factor;

         }////////end of
function loop

//////////////////////////

//////////////////////////
 void rpm()
     {
         rpm_counter++;


     }

//////////////////////////////
/////

//////////////////////////////
////
     void Digits_Display()
 {
     for(int j=size; j>=0; j-
-)
     {
         data=sdigits[j];
         datawrt();
         }
    }//////////////end of
function


     /*int DecimalToAscii(int
x)
          {
              int y;
              if(x==0)
              y=48;
              if(x==1)
              y=49;
              if(x==2)
              y=50;
              if(x==3)
              y=51;
              if(x==4)
              y=52;
              if(x==5)
              y=53;
              if(x==6)
              y=54;
              if(x==7)
              y=55;
              if(x==8)
```

```
                y=56;
                if(x==9)
                y=57;


                return y;
     }*/ ////////////end of
function

////////////////////////////
//////
    void separate_digits(int
y)
     {
        size=0;
        do{

sdigits[size]=DecimalToAscii(
y%10);
         y=y/10;
         size++;
         }while(y!=0);
         size=size-1;
///////due to one extra count
     }


////////////////////////////
////////////////////////////
/////////////////


////////////////////////////
////////////////////////////
/////////////////
     void display()
    {
```

```
              Matlabdata =
(rpm_counter*60)/2;
              printf("%f\n",
Matlabdata);
              Save_RPM =
Matlabdata;
             if(Ramp_Time ==
0)
              {

                   j = 1;
                   ledr = 1;
                   }


             else
             {
                 new_reference
= new_reference +
Ramp_Fraction;/////if
previous is 2100 next will be
2100 + increment
              }


              data=0X01;
              comandwrt();
              data=0X0E;
              comandwrt();
              data=0X06;
              comandwrt();
              data=0X38;
              comandwrt();
              data=0X85;
              comandwrt();
```

```
            Ramp_Stage();


separate_digits((rpm_counter*
60)/2); /////because we are
taking 1 after one second
reading

RampRPM_Display();
            data=0XCD;
            comandwrt();
            Digits_Display();


RampTime_Display();

separate_digits(Ramp_Time);
            Digits_Display();

//////////////////////13th

            rpm_counter=0;
            Ramp_Time =
Ramp_Time -1;


        }

//////////////////////////////
//////////////////////////////
///////////

//////////////////////////////
//////////////////////////////
//////////
        void display1()
                {

                if(SS_Time == 0)
                {

                    j = 1;

                }

////////////////////
                Matlabdata =
(rpm_counter*60)/2;
                printf("%f\n",
Matlabdata);

////////////////////
                data=0X01;
                comandwrt();
                data=0X0E;
                comandwrt();
                data=0X06;
                comandwrt();
                data=0X38;
                comandwrt();
                data=0X85;
                comandwrt();

                SS_Stage();


separate_digits((rpm_counter*
60)/2); /////because we are
```

```
taking 1 after one second
reading


            SSRPM_Display();
            data=0XCA;
            comandwrt();
            Digits_Display();
            data=0X94;
//////must be on third line
            comandwrt();
            SSTime_Display();

separate_digits(SS_Time);
            Digits_Display();




            Sensor_value =
(rpm_counter*60)/2;
            Error = SS_RPM -
Sensor_value;
            Error_Sum =
Error_Sum + Error;

if(Error_Sum>4000)
            Error_Sum =4000;
             P = Kp*Error;
             I =
Ki*Error_Sum;
             D = Ki*(Error-
Last_Error);
             PID = P+I+D;


             if(PID>=12500 ||
PID<=-12500)
                PID = 12500;


                PID =
PID/Divider;
                if(PID>0)
                    myservo =
PID;
                if(PID<0)
                    myservo = -
PID;
                ss = 1;
                Last_Error =
Error;

//////////////////////
             rpm_counter=0;
             SS_Time = SS_Time
- 1;


         }
////////////////////////////
////////////////////////////
////////////
////////////////////////////
////////////////////////////
////////////////////////////
///////
void system_settings()
{
             data=0X01;
             comandwrt();
             data=0X0E;
             comandwrt();
             data=0X06;
```

```
comandwrt();
data=0X38;
comandwrt();
data=0X85;
comandwrt();
data='W';
datawrt();
data='a';
datawrt();
data='i';
datawrt();
data='t';
datawrt();
data='.';
datawrt();
data='.';
datawrt();
data='.';
datawrt();

data=0XC0;
comandwrt();

data='H';
datawrt();
data='i';
datawrt();
data=',';
datawrt();
data=' ';
datawrt();
data='w';
datawrt();
data='e';
datawrt();
data=' ';
datawrt();
data='a';
datawrt();
data='r';
datawrt();
data='e';
datawrt();
data=' ';
datawrt();

data=0X94;
//////must be on third line
comandwrt();

data='g';
datawrt();
data='e';
datawrt();
data='t';
datawrt();
data='t';
datawrt();
data='i';
datawrt();
```

```
data='n';
datawrt();
data='g';
datawrt();
data=' ';
datawrt();
data='t';
datawrt();
data='h';
datawrt();
data='i';
datawrt();
data='n';
datawrt();
data='g';
datawrt();
data='s';
datawrt();


data=0XD8;
comandwrt();


data=' ';
datawrt();
data='r';
datawrt();
data='e';
datawrt();
data='a';
datawrt();
                data='d';
                datawrt();
                data='y';
                datawrt();
                //Hi, we're
getting things ready for you
    }
////////////////////////////
////////////////////////////
//////////////////////
////////////////////////////
////////////////////////////
//////////////////////
int main()
  {
    c1.mode(PullUp);
    c2.mode(PullUp);
    c3.mode(PullUp);
    c4.mode(PullUp);
    start = 1;

////////////////////////////
    system_settings();

////////////////////////////
    ///////ESC
Settings///////////////
    myservo = 0.0;
    ledf = 1;
    wait(0.5); //detects
signal
    //Required ESC
Calibration/Arming sequence
```

```
    //sends longest and
shortest PWM pulse to learn
and arm at power on
    myservo = 0.13; //send
longest PWM
    ledf = ledr = 0;
    wait(8);
    //myservo = 0.0; //send
shortest PWM
    //wait(8);



    while(1)
    {
              r1=0;
              r2=0;
              r3=0;
              r4=0;


              data=0X01;
              comandwrt();
              data=0X0E;
              comandwrt();
              data=0X06;
              comandwrt();
              data=0X38;
              comandwrt();
              data=0X86;
              comandwrt();


              ///////////////
```

```
                  data='E';
                  datawrt();
/////////////Wellcome
                  data='n';
                  datawrt();
                  data='t';
                  datawrt();
                  data='D';
                  datawrt();
                  data=' ';
                  datawrt();
                  data=':';
                  datawrt();
                  data=' ';
                  datawrt();



                  spincoater();
                  while(data!=65)
                  {


                     if(c1==0 |
c2==0 | c3==0 | c4==0)
                      {

data=keypadN();
                         wait(0.5);

if(data==65)
```

```
                    {

RampRPM();

wait(0.001);

timeup.attach(&display,1);

sensor.rise(&rpm);/////ISR

NVIC_SetPriority(EINT2_IRQn,
1);

////////////////////////////
/////code from sspid
section////

////////////////////////////
//////////////////////

                        //SS_RPM
= 5500;

//SS_Time = 30;

data=0X01;

comandwrt();

data=0X0E;

comandwrt();

data=0X06;

comandwrt();

data=0X38;

comandwrt();

data=0X85;

comandwrt();

Ramp_Stage();



//Ramp_RPM = 5000;

//Ramp_Time=10;


                        start =
0; /////start led off

if(SS_RPM>6000)
                        SS_RPM =
6000;


                        int
scale = SS_RPM/30; ///////e.g
4000/30 = 133 instead of
133.33
                        SS_RPM =
scale*30; ///////133*30 =
3990



if(Ramp_RPM>6000)
                        Ramp_RPM
= 6000;  ////putting limit
for max rpm

if(Ramp_RPM>=5700)
```

```
                    {
Ramp_RPM = Ramp_RPM + 30;
///////adding offset
                        }

if(Ramp_RPM>=5300 &&
Ramp_RPM<5700)
                        {

Ramp_RPM = Ramp_RPM + 100;
//////adding offset
                        }


Ramp_Fraction = (Ramp_RPM-
2800)/Ramp_Time;
//////because motor starts
from 2000 rpm
                    factor =
0.047;
                    myservo
= factor;
                    ramp =
1;


                    //int
i=0.5;
                    int
local_rpm=0;
                    j = 0;

new_reference = new_reference
+ Ramp_Fraction; ////////////
                    while (j
== 0)
                    {
                    wait(1);

local_rpm = rpm_counter;

Error = new_reference -
local_rpm;

Error_Sum = Error_Sum +
Error;

if(Error_Sum>4000)

Error_Sum =4000;
                                P =
Kp*Error;
                                I =
Ki*Error_Sum;
                                D =
Ki*(Error-Last_Error);
                                PID
= P+I+D;


if(PID>=12500 || PID<=-12500)
                                PID
= 12500;

                                PID
= PID/Divider;
if(PID>0)

myservo = PID;

if(PID<0)
```

```
myservo = -PID;


Last_Error = Error;

new_reference = new_reference
+ Ramp_Fraction;


                  }
                  ramp = 0;
////ramp led of
                  //////end
of while
          // __disable_irq();

//NVIC_DisableIRQ(TIMER2_IRQn
);

timeup.detach();
                  //factor =
factor - 0.002677419;
                  //myservo =
factor;

//////////////////////

over.attach(&display1,1);
                  ledr = 0;
                  j = 0;
                  rpm_counter
= 0;

wait(SS_Time+1);

over.detach();
                        ledr = 1;
                        myservo =
0;

//RampCalculations(); ////

//////////////////////////////
//////

//////////////////////////////
//////

//RunTimeDisplay2();
                           ss = 1;
                           start =
1;
                           ramp =
1;

Process_Complete();

rpm_counter = 0;
                           ss = 0;
                           ramp =
0;
                           start =
1;
                     }


               }///end of
keychecker if
             }  //end of
internal while
            }//////////end
of while
```

```
}///end of main
```

References:

1. 'Spin coating', 2017. [Online]. Available: https://www.ossila.com//spin-coating . [Accessed: 3-Feb- 2017].

2. 'Manual Spin Coating', 2017. [Online]. Available: https://www.spincoating.com/en/applications/manual-coating-spin- coater/36/ Accessed: [3- Feb- 2017].

[3]- "Embedded Systems - Interrupts" [Online]. Available https://www.tutorialspoint.com/embedded_systems/es_interrupts.htm . [Accessed: 08-Jan-2018]

[4]- Liu, Commonly Used Approaches to Real-Time Scheduling, Real Time Scheduling Theory, Chapter#04, Lund University Texas, USA, November 11, 2013.

[6]- 'Wafer Spin processor' 2017 [Online]. Available: http://www.sparetech.com/POLOS_Serie_Single_Wafer_Spin_Processor_Brochure Accessed: 6- Feb- 2017].

[7]- M. F. Hossain, "Fabrication of Digitalized Spin Coater for Deposition of Thin Films", IEEE xplore online magazine, pg 1-4

[8]- Arshad Hassan,Shawkat Ali, Gul Hassan, "Inkjet-printed antenna on thin PET substrate for dual band Wi-Fi communications" ,Microsyst Technol , Berlin Heidelberg, 17 August 2016

[9]- Haider Raad Khaleel "Flexible printed monopole antennas for WLAN applications" ,Microsyst Technol , Kentucky USA, July 2011.

[10]- K.W., "A Direct-Write Printed Antenna on Paper-Based Organic Substrate for Flexible Displays and WLAN Applications," Display Technology, Journal of , vol.6, no.11, pp.558-564, Nov. 2010.