Smart Ring Spinning Optimization in Textile Industry

Submitted by:

Usama Amir 2017-EE-43

Saqib Nisar 2017-EE-45

Muhammad Suleman 2017-EE-166

Muhammad Usman Khan 2017-EE-187

Supervised by: Sir Ubaid Ullah Fayyaz

Co-supervised by: Mr Umer Shahid

Department of Electrical Engineering

University of Engineering and Technology Lahore

Smart Ring Spinning Optimization in Textile Industry

Submitted to the faculty of the Electrical Engineering Department
of the University of Engineering and Technology Lahore
in partial fulfillment of the requirements for the Degree of

Bachelor of Science

in

Electrical Engineering.

Internal Examiner

External Examiner

Director
Undergraduate Studies

Department of Electrical Engineering

University of Engineering and Technology Lahore

Declaration

We declare that the work contained in this thesis is my own, except where explicitly stated otherwise. In addition this work has not been submitted to obtain another degree or professional qualification.

Name: Usama Amir
Signed: _____
Date: _____

Name: Saqib Nisar
Signed: _____
Date: _____

Name: Muhammad Suleman
Signed: _____
Date: _____

Name: Muhammad Usman Khan
Signed: _____
Date: _____

Acknowledgments

Thanking to Sir Ubaid Ullah Fayyaz for their guidance in the completion of project.

Dedicated to the Faculty of Electrical Engineering UET Lahore...

Contents

List of Figures

Abbreviations

RPM	**R**evolution per minute **P**er **M**inute
PCB	**P**rinted **C**ircuit **B**oard
IR	**I**nfra **R**ed
BLDC	**B**rush**L**es **D**irect **C**urrent

Abstract

Optimization of yarn breaking detection can play a very significant role in the textile industry. Industry loss huge amount of money and much amount of yarn bobbins waste due to breaking of thread. Thus if this issue can be solved or optimized by detecting the thread breaking on time, industry can be benefited with healthy profit. For staple yarn production, most common measuring method to characterize the dynamic yarn path in the ring spinning process is to measure the yarn tension, where the yarn path is almost straight. However, it is much more complex to measure the yarn tension at the other positions. The project aims at providing an efficient solution to optimize thread break detection. Some of those researches are analyzed and reviewed. The main aims and objectives of the project have been presented and how the development of the project and its methodology is going to be conducted is described. All the necessary block diagrams and flowcharts and the designated hardware equipment have also been described.

Chapter 1

Introduction

In the textile industry, in view of the spinning machine in the spinning process in a sudden disconnection situation and difficult to be found. It is necessary for the operator to use the eyes back and forth to inspect the test. But this way labor intensity, also prone to visual fatigue, and will cause misjudgment of broken yarn. If the yarn cannot be detected and repaired in real time on the spinning frame. Some parts of the spinning machine may be malfunctioning, and resulting in the spinning frame cannot work properly, even damaged. And the spinning machine is the key equipment in the entire production chain, once damaged, will directly cause the textile mill spinning system cannot be effectively run, resulting in economic losses. Therefore, the spinning yarn breakage detection technology in the modern textile production process has been more and more attention. The tensions in the yarn during ring spinning can be considered with respect to mainly three zones:

1.1 Yarn formation zone (region I):

From the nip of the delivery rollers of the drafting system to the yarn guide, the yarn tension is termed as the spinning tension and is governed mainly due to the balloon tension at the yarn guide.

1.2 Balloon zone (region II):

From the yarn guide to the traveler, tension occurs in the yarn length referred to as the balloon tension (Tb) at a given point on the balloon length, varying with amplitude measured from the spindle axis. The yarn in region II gives rise to dynamic forces, which determine the level of tension in the yarn and its distribution along the yarn path.

1.3 Winding zone (region III):

In the yarn length from the traveler to the winding point onto the cop, the yarn tension is termed as the winding tension (Tw). This yarn part is rotated according to the spindle axis on the ring-traveler system.

Chapter 2

Problem Statement

Today's textile markets are highly competitive throughout the entire value chain fiber to fabric. Customers expect unique products of the right quality and free from unacceptable defects every time. Mills need to manufacture economically, with the best possible use of resources, especially in raw materials and labor. These are the challenges requiring comprehensive mill management strategies.

The main problem concern in the textile Industry is a thread break while bobbin buildup. This end break detection is only possible if a staff member detects that end breaks as soon as possible. Due to the end break, there is no control over the yarn delivery rollers unless the staff member figure that out. Moreover, the waste yarn spindle due to undesirable end break was unacceptable. These unfortunate happenings are the major causes of a new beginning.

Maximizing profitability where it matters most. The product that would be made in this project will measure and control the critical parameters in ring spinning. Mill conditions, machine speeds, parts and maintenance, and personnel reaction times are crucial for profitability. End-break detection with our product is the starting point for best-practice in optimization.

Chapter 3

Literature Review

3.1 Measuring techniques of dynamic yarn tension in a ring spinning process

Available Solution

At present, the ring spinning process is the most widely used spinning method for yarn production in the textile industry. The most easy and common measuring method to illustrate the dynamic yarn tension in the ring spinning process is to measure the yarn tension, where the yarn path is quite straight. It is relatively much difficult to measure the yarn tension at other possible positions. For example, balloon zone (path from yarn guide to traveler) and winding zone (path from traveler to winding point).[1]

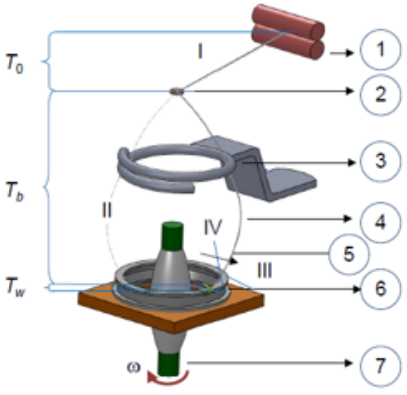

FIGURE 3.1: Principle of ring spinning and definition of different regions in the yarn path; T0 spinning tension, Tb balloon tension, Tw winding tension,(I) between delivery rollers and yarn guide,(II) between yarn guide and traveler, (III) through traveler, (IV) between trave-ler and winding point of cops,(w) angular velocity of spindle, (1) delivery rollers, (2) yarn guide, (3) balloon control ring, (4) yarn balloon, (5) ring, (6) traveler, (7) spindle.

In this regard, FIBRES AND TEXTILES IN EASTERN EUROPE, 2016 came up with the two optimized methods for the measurement of yarn tension in the balloon zone.

3.2 Method 1

The balloon shape was initially recorded with a high-speed camera and then the balloon tension was calculated by comparing yarn strain (measure by digital image analysis program in balloon zone) with stress-strain curve of the yarn.

3.3 Method 2

The radial forces of the rotating balloon were measured by using yarn tension measurement technique. Furthermore, a customized sensor was developed to measure the winding tension between traveler and the spindle.

Disadvantages

Measurements were performed at different spindle speeds such as 5,000, 10,000 and 15,000 r.p.m. A good correlation was observed between measured and the calculated values.

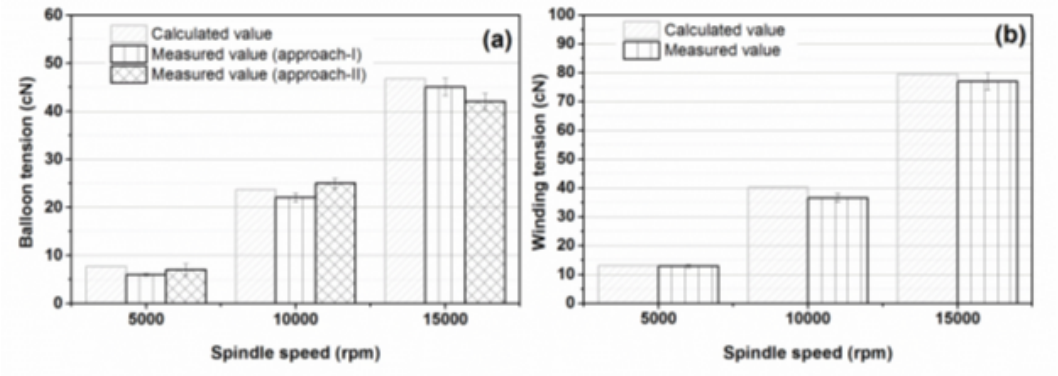

FIGURE 3.2: Spindle Rpm chart

However, the study of values conclude that the measured values came out to be smaller than the calculated ones.

- The first reason for the difference is the influence of different amount of twists due to force distribution along the balloon length.
- The second reason is the dynamic friction occurring between the rotating balloon and measuring sensor.

3.4 Rotational speed measurement of ring spinning based on magnetic sensor

Available Solution

Rotational speed measurement of the ring spinning is one of the most significant parameters of the textile industry. This parameter can contribute to every required output in the calculations and detection in the ring spinning process.[2]
If we observe the working relation of the traveler along the spindle, we would come to the conclusion that the drag in the speed of traveler is due to the friction between the ring and the traveler. Secondly, due to the resistance between the yarn and the air. This resulting speed difference between spindle and the traveler is the cause of twist in the yarn.
[3]

Traveler speed and yarn breaks caused by tension, affects the production.
Rotation speed of traveler will fluctuate during ring spinning process due to

- Slippage between spindle and belt.
- Vibration of different elements.
- Instability of spindle speed.

So real time monitoring helps in detection of:

- Rpm of the traveler
- Yarn breakage (as traveler runs slow and eventually stops because of friction between traveler and ring when yarn breaks)

3.5 Methods already implemented

There are two available methods to measure rotational speed of traveler during the ring spinning operation include:

3.5.1 Photoelectric technology

This technology uses a reflective light to determine position of rotating traveler. Rotational speed can be measured by processing of light signal detected by photoelectric detector.

3.5.1.1 Disadvantage

Its accuracy is less as it is sensitive to the flying fiber.

3.5.2 Electromagnetic effect-based detection

When traveler with high rotational speed cuts the magnetic line of force of static magnetic field, induced voltage is induced due to change in magnetic flux. Speed can be calculated from the signal result of induced voltage.

3.5.2.1 Disadvantage

- Induced voltage attenuates with the increase of detection distance.
- Output signal is sensitive to fixed angle and rotational speed.

3.6 New Method

A new rotational speed measurement method based on magnetic anomaly detection was proposed by MEASUREMENT SCIENCE AND TECHNOLOGY in 2017 to monitor traveler's speed in ring spinning. Rotational speed can be calculated from the output signal acquired from tunnel magneto-resistance.[4].

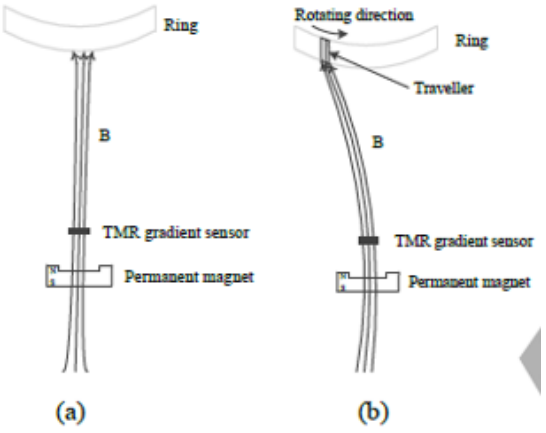

FIGURE 3.3: Schematic diagram of structure and operation principal of magnetic sensor

3.6.1 Magnetic Anomaly Detection

Magnetic anomaly caused by ferrous or magnetic object motion can be used to locate the targets. Periodic disturbance caused by rotation of ferrous traveler with some speed will produce regular signal. Rotational speed can then be calculated by signal processing. The anomalous magnetic signal can be detected be field sensitive magnetoresistance device available in markets e.g.

- Anisotropic magnetoresistance (AMR)
- Giant magnetoresistance (GMR)
- Tunnel magnetoresistance (TMR)

Chapter 4

Methodology

4.1 Background

The main idea behind this project is to measure the rpm of the traveler which gives vital information about the system. Real-time monitoring enables the status of individual spindles to be checked and information such as yarn length, yarn production, and production efficiency can be extracted.[2]

4.2 Why measure RPM?

During spinning, the roving is attenuated to the desired fitness while the front roller rotates fast and the back roller stabilizes the incoming thread. The yarn then passes through the thread guide, which is threaded to the bobbin through the traveler that circularly runs along the ring. The spindle rotating at high speed drives the rotating traveler through the tension of yarn. The rotation speed of traveler will fluctuate during the ring spinning process for various reasons, such as slippage between the spindle and the belt, vibration of different elements of ring spinning machine, and instability of spindle speed. The unstable speed of traveler steeply rises at the end breakage rate for the fluctuation of tension. The real-time detection of yarn break can be realized from the speed measurement of traveler based on the fact that the traveler runs slow and eventually stops because of the friction between the traveler and the ring when the yarn breaks.

4.3 Method/Procedure:

An IR sensor is placed in small gap from spindle which detects the traveler in each revolution and a signal is generated. This signal is amplified and noise is removed using filter. A schmit trigger is designed to attain pulses at two thresholds which constitutes the pulses. These pulses are fed to microcontroller and processed to calculate the rpm of traveler. As the rpm reduces to zero, this indicates that yarn has broken. The real-time monitoring of the rpm can be utilized to extract useful

information.

Now for the connection between the microcontroller and yarn-spindles, RS-485 communication protocol is used. All the senders on the RS485 bus are in tri-state with high impedance. In most higher level protocols, one of the nodes is defined as a master which sends queries or commands over the RS485 bus. All other nodes receive these data. Depending of the information in the sent data, zero or more nodes on the line respond to the master. In this situation, bandwidth can be used for almost 100%. There are other implementations of RS485 networks where every node can start a data session on its own. This is comparable with the way ethernet networks function.

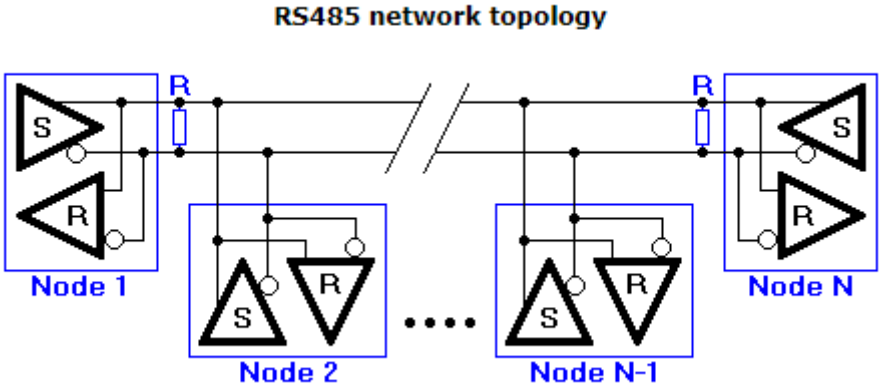

FIGURE 4.1: RS485 Networking [2]

Chapter 5

Detail System Design

5.1 Simulation

The first major component of the project is to detect the either the thread is break in the yarn or not. For this a circuit is made to detect the break thread. There are three major stages in the circuit:

- Stage 1: Signal conditioning circuit
- Stage 2: Inverting amplifier circuit
- Stage 3: Schmitt Trigger circuit

5.1.1 Signal Conditioning circuit

It has a further three components:

- Common Emitter circuit
- Comparator circuit
- Low Pass filter

The Proteus simulated circuit of signal conditioning circuit is shown below:

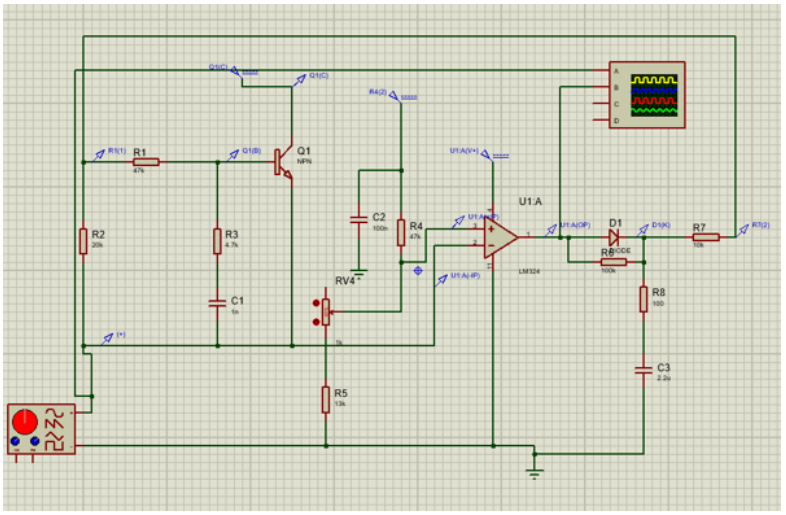

FIGURE 5.1: Stage 1 of circuit

5.1.1.1 Common Emitter circuit

At extreme left of figure the first part is common emitter amplifier component. The purpose of this part is to increase the strength of the signal in terms of the voltage that is received from the IR sensor.

5.1.1.2 Comparator Component

After common emitter component, there is a comparator component in the middle part of figure. As it can be seen, the non-inverting terminal is permanently biased, it has a voltage of 0.52 volts. The comparator will compare the two signals that either the signal that we have received from the IR sensor is a noise or the thread is actually break.

5.1.1.3 Low Pass Filter

At last, there is a third component i.e. low pass filter. The purpose of this low pass filter to remove the any type of noise or frequencies or unwanted frequencies. But before that, you can see there is our diode and resistor. The purpose of this is to detect either the thread is break or not. When the voltage is more than 0.7 from the comparator then the diode will be short and it will show the output but if the voltage is no more than 0.7 then it will pass through the resistor to limit the voltage and then the output will be shown. The important thing here is that, there is a feedback from the output to the base part of the common emitter part. This feedback is again testing the signal that either in the next part thread is break and if it is break, then it will show and it will be produce square waveform. All the values of capacitors and resisters are standard. Only at the non-inverting terminal, the value of resistor is selected to attain a voltage of 0.5V. Except that all the values are standard.

5.1.2 Inverting Amplifier circuit

After stage 1, a complete square waveform is obtained that is of varying frequency and from this the speed of the spindle or bobbin that is moving on the motor can be measured. Moving towards the stage 2 there is an inverting amplifier circuit. The output that is obtained from the stage 1 have a distortion at particular high frequencies. So, to make it compatible for higher frequencies or in a technical way saying at a higher speed, there is an inverting amplifier circuit. The purpose of this amplifier circuit is to amplify the signal that is coming from the output of stage 1. It will amplify it and then after it there is a standard low pass filter to remove or high order sequences.

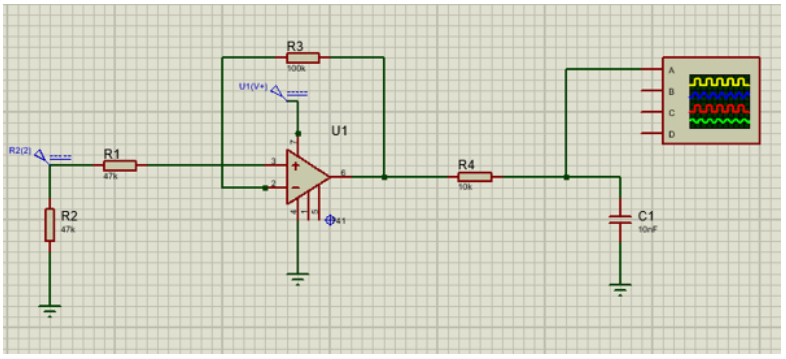

FIGURE 5.2: Stage 2 of circuit

A Vcc of 5V and a gain of 2.23 will be obtained and then multiplying it with input voltage (i.e. the output of stage 1) it will be almost 10V signal.

5.1.3 Schmitt Trigger circuit

After inverting amplifier, there is Schmitt trigger circuit. The output waveform that is obtained at a stage 2, work fine at higher frequencies but it has a flaw that the waveform was bit distorted i.e. the waveform was not square shaped. So, to obtain a square wave at the output of varying frequency Schmitt trigger circuit is used.

For Schmitt trigger circuit two values are required: Upper threshold level and the lower threshold level. it is a range in which we want to obtain a square wave.

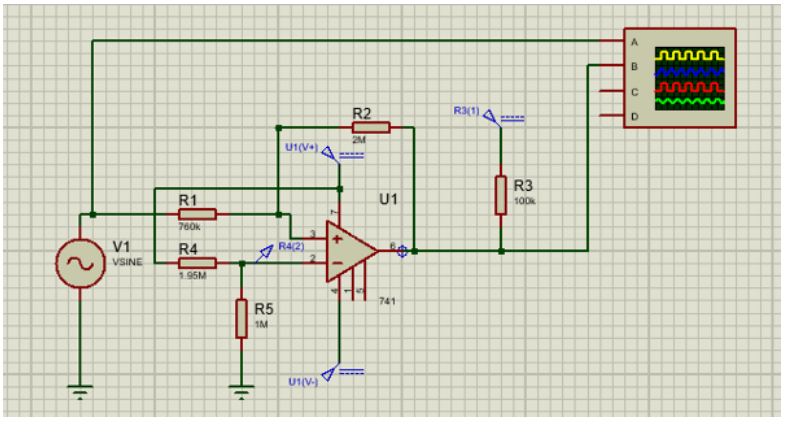

FIGURE 5.3: Stage 3 of circuit

The region in which there is a clean square waveform is observed from the oscilloscope and that are the upper and lower threshold values respectively. In the figure below, the circuit diagram of Schmitt trigger circuit simulated on the Proteus is shown. Only the value of resistor R1 and R4 controlling the upper and lower threshold level. The calculations for R1 and R4 are described next. Apart from that, all the values of resistors are standard and Vcc of 5V is given as conventional. The important thing is that there is a pull up resistor attached to output. The purpose of this pull up resistor is to control the level of current. Since the voltage is very high at that output thus the current should be controlled so that it could not burn the motor. So, the pull up resistor is used.

5.1.3.1 Calculations of Schmitt Trigger circuit

For Schmitt Trigger Circuit, We have following values and require following values of resistors and voltages.

Sr. No	Components	Values
1	Lower Threshold Voltage at stage 2 (V_L)	0.5 V
2	Upper Threshold Voltage at stage 2 (V_H)	2.3 V
3	Standard Value of Resistor R_2	2MΩ
4	Standard Value of Resistor R_3	100kΩ
5	Standard Value of Resistor R_5	1MΩ
6	Require Value of Resistor R_1	?
6	Require Value of Resistor R_4	?

TABLE 5.1: Component Values for Schmitt Trigger Circuit

Formulas:

$$V_T = (\frac{R2}{R1 + R2})V_H \tag{5.1}$$

$$V_T = \frac{2}{R1 + 2}(2.3) \tag{5.2}$$

$$V_T = (\frac{R2 + R3}{R1 + R2 + R3})V_L + (\frac{R1}{R1 + R2 + R3})V_p \tag{5.3}$$

$$V_T = \frac{2.1}{R1 + 2.1}(0.5) + \frac{R1}{R1 + 2.1}(5) \tag{5.4}$$

Subtracting eq. (5.2) and (5.4) and by further simplifying we get:

$5\mathrm{R}_1^2 + 6.45 \times 10^6 - 7.56 \times 10^(12) = 0$

$\mathrm{R}_1 \cong 760k\Omega$

Putting value of $\mathrm{R}_1 ineq.(5.1)$:

$V_T = 1.68V$

From the circuit, it can be observed that:

$$V_T = (\frac{R5}{R4 + R5})(V_r) \tag{5.5}$$

Solving this eq. for R_4 we get:

$R_4 \cong 1.95M\Omega$

Thus required values (standard) of resistors are:

$R_1 = 760k\Omega$
$R_4 = 1.95M\Omega$

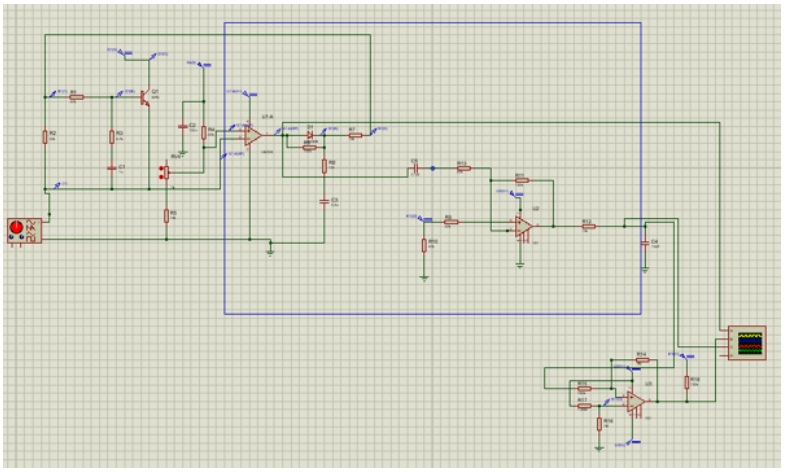

FIGURE 5.4: Complete Circuit Simulation

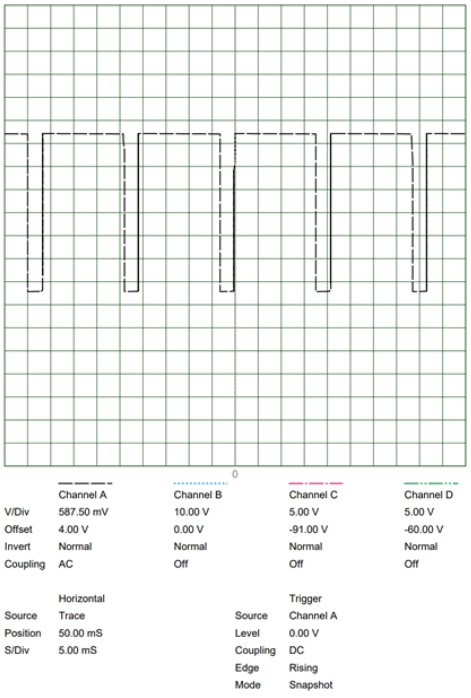

FIGURE 5.5: Output Waveform

Chapter 6

Hardware Developed

6.1 Motor Setup

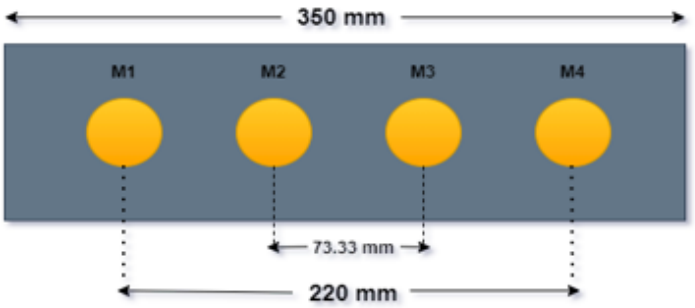

FIGURE 6.1: Motors Setup

There are four motors on a wooden patch of length 350mm. consecutive motors are spaced 73.33mm apart each. and the length from first to the last motor is 220mm.

6.2 Prototype Setup

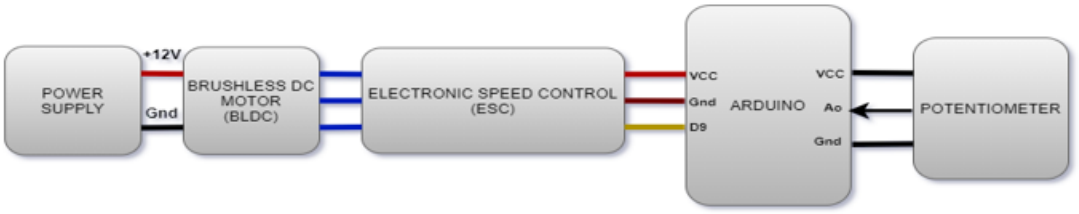

FIGURE 6.2: Prototype Setup

The components of this prototype are as follows:

- Power Supply
- Brushless DC Motor(BLDC)
- Electronic Speed Control(ESC)
- Arduino
- Potentiometer

6.2.1 Power Supply

The 12V DC power supply has been used to power up all four brushless dc motors(BLDCs).

6.2.2 Brushless DC Motor

- A2212/ T10 1400kV BLDC Motor
- Max. Efficiency Current: 6-12A
- Max. Watts: 180W
- Max Efficiency: 80

FIGURE 6.3: BLDC [1]

As the name suggests, it is a Brush less dc motor, so it has no brushes. the main reason we want for our product is its long life. The brushes in the motor wear out very soon. this is lighter in weight, less noisy providing higher efficiency at low cost. its three phase connections assist in controlling the motor speed smoothly.

6.2.3 Electronic Speed Control

This is a 30A ESC. It's an electronic circuit that controls and regulates the speed of an electric motor.

FIGURE 6.4: ESC [5]

It is also known as Battery Eliminator Circuit. As the name suggests, there is no external battery required to power up arduino, thus, the regulated 5V are supplied to arduino by Electronic speed controller.

6.2.4 Arduino

Arduino gets the signal from potentiometer and the provides the required signal to ESC for speed control of the motor. Arduino pro mini has been used for this purpose.

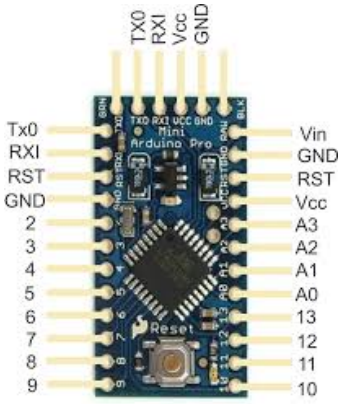

FIGURE 6.5: Arduino

Ao pin of arduino was used for potentiometer and D9 pin was used to provide signal to ESC module.

6.2.5 Potentiometer

Potentiometer of 10k has been used to vary the speed of motors.

FIGURE 6.6: Potentiometer

6.2.6 Setup

This was the final setup of connection of each module with the other module.

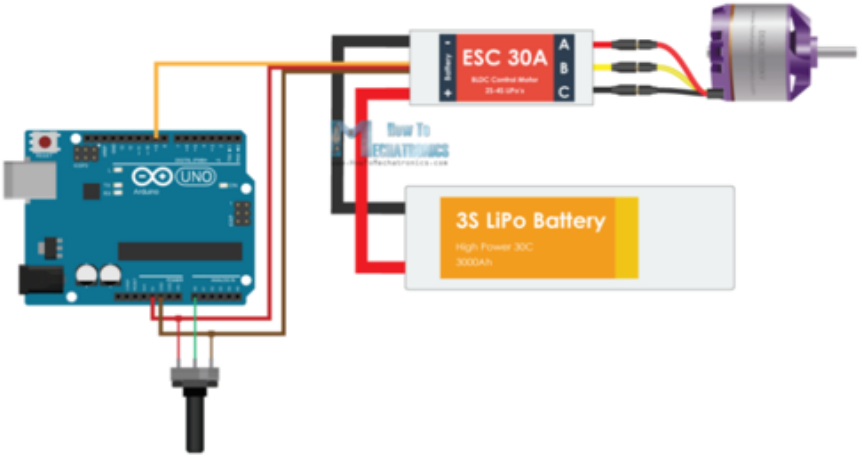

FIGURE 6.7: Setup [6]

6.3 Testing

The final step in hardware was final testing of the prototype.

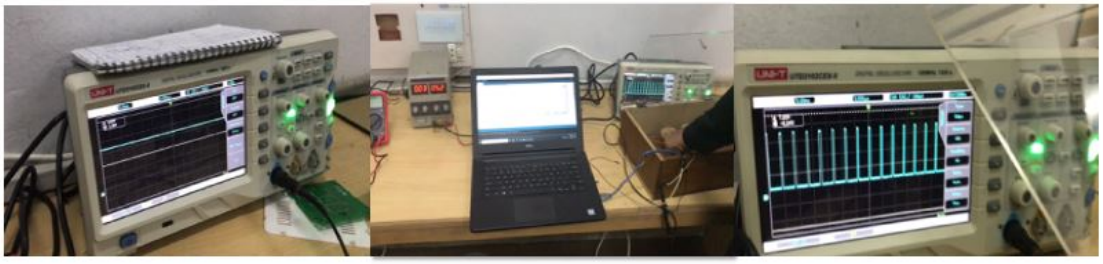

FIGURE 6.8: Final Testing

The circuit was patched, motor was started and the Arduino codes helped us find the frequency and rpm of the spinning prototype.

Chapter 7

Software Implementation

7.1 Controlling BLDC Motor

For testing purpose, we have used BLDC motors with such caps that would act as moving thread in the spindles. BLDC motor runs at a desired speed that can be control by potentiometer. Also, BLDC motor driver is used that operate from the guideline of Arduino code. See Appendix A1.

Code is written in a well manner and is explained with the help of appropriate comments. This piece of code use pre-exists Arduino Servo library that can be used to import function for BLDC motor which would control the motor according to requirement.

7.2 Measuring RPM of Motor

IR sensor is used to detect the changing in the RPM of motor. IR sensor is placed next to thread at a particular distance. This IR sensor provide Analog value to our controller that is Arduino UNO. This Analog value is then converted to digital value and that digital value is used to define the RPM of our spindle in our prototype of our servo motor. Given is the Arduino that is used in our project to measure the RPM value of our spindle. Since our one protype is getting data from spindle at a time so an array of size 4 is used to store or display that data. See Appendix A2.

Here is the setup code of RPM measurement code. We have tried to explain each line of code with the help of comments so that one could easily intercept code just by looking. In this piece of code at first, we are defining pin 8 as our output and setting it to low. Then All 4 pins 4,5,6 and 7 are set to low before reading any input from IR sensors. SREG (status register) store the current value of system

interrupts and then analog comperator are used to compare the analog value and also after then these analog values are converted to digital values.

In this portion of code, we have first enabled the analog comparator. ADMUX values are determined and then these bits are used to select particular analog input channel. After converting these analog values into digital, these frequency values are stored in a n array and then displayed to a serial monitor. At the end whole step is repeated and we get RPM of each spindle continuously from 1st spindle to 4th spindle. If value is greater or less than a specific range interrupts are generated.

7.3 Master, Slave Communication

Master, Slave communication can be easily explain with the help of below diagram.

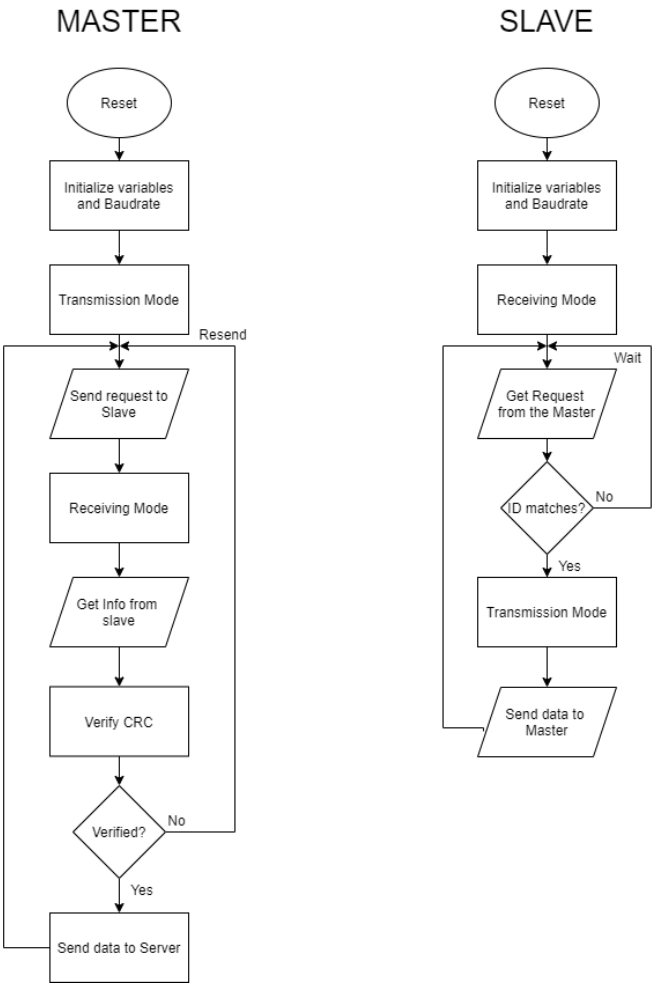

FIGURE 7.1: Master Slave Communication

7.3.1 Master Code

At start we have imported required libraries in our master code. Time library provide us the basic function of time and give us functions like hours(), minutes(), seconds(), day(),weekday() etc.. Crc16 library is used to avoid cyclic redundancy error detection. SoftwareSerial library give us the functionality to use our digital pins for serial communication purpose. After importing required libraries we have defined required variable to use in our preceding code. See Appendix A3.1 for Master Code.

A class request is being created that would be called whenever our master would want to request slave to send data. This crequest class is simple and have only two defined variable as id and fc. The setup code describe our controller about the pin usage and their specification like whether used as input or output. Also, Enable pin is set to low initially so we won't have any arbitrary data at start. Below is loop code of our Master controller. Initially we clear the Crc by ClearCrc() function. Master pin is enabled and request is send to slave to send data. After that transmission enable pin is set to low to put it in an receiving mode to received data from called slave. Then this data is displayed on serial monitor.

7.3.2 Slave Code

Initially required variable are defined that would be used in our coding. Also, Crc16 library is imported so that we could detect any cyclic redundancy in our data and check for any accidental change in in our data. SoftwareSerial is also imported to use our digital pin for serial communication. In our demo code we have used 4 slave and each slave has RPM of 4 spindle that is selected randomly. See Appendix A3.2 for Master Code.
Setup include defining pin 7 as an input pin. After that we have saved interrupt register data into SREG before using it for our purpose. Slave enable pin is set to low initially so at start we wont be transferring or receiving any arbitrary data. Baud rate is selected as of 9600.

Decision function is called in our loop code that get id as input and then used that id to provide rpm value of spindle present in a certain slave. This decision function also use ClearCrc() function initially to clear any redundancy and then update it with demo value of rpm. Slave Enable pin is set to High to put it as a sender and then rpm data is send to master.once all data of 4 spindle is send enable is set back to low in a receiving mode.

7.3.3 Implementation

The rpm of each spindle is measured and calculated by slave. There are four spindle attached to one slave and 24 such slaves are connected to one master. Each slave has a unique ID that master uses to address the particular slave. The

master acts according to data received from slave and notify the operator. As the system boot-up, the slave starts to take input from the IR sensors. Master sends an ID to Modbus, i.e. the bus where all the slaves are connected. The slave whose ID matches with the requested ID responds to the master request in accordance with the function code sent by master along with the request. The master then verifies the received data for noise by applying Cyclic-Redundancy-Check (CRC) and upon successful authentication moves to next slave. This occurs every 0.5s.

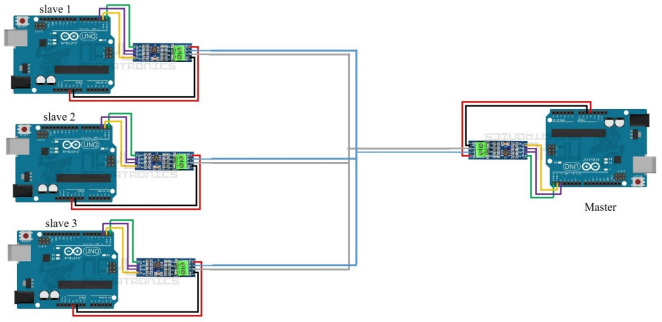

FIGURE 7.2: Implementation of master and slave

Chapter 8

Results

We have made prototype to measure the RPM of Moving spindles. Setup and Result of that prototype is shown.

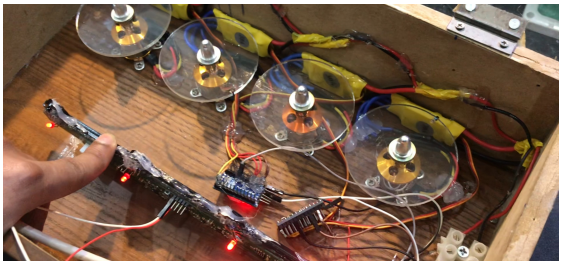

FIGURE 8.1: Prototype

Following result are obtained from 4 moving spindles. As you can see first 3 spindles give the RPM values and these values are not much different from actual values. We have removed the traveller from 4th spindle so it act like a broken thread and that's is the reason we are getting 0 RPM value at that end. results are obtained at various RPM of motor and conclusively we can say our prototype is working fine at different values of RPM.

	entry	spindle_1	spindle_2	spindle_3	spindle_4	rec_time
▶	1	123.3	253.2	147.5	248.6	2021-02-15 01:25:55
	2	458.6	247.5	125	147.3	2021-02-15 01:26:10
	3	555	478.1	447.6	906.2	2021-02-15 01:26:40
	4	547	457.4	741.3	455.2	2021-02-15 01:27:00

FIGURE 8.2: Output obtained

Chapter 9

Future Work

9.1 Master to R-pi Communication

The master now has data for all slaves and is ready to render it to cloud but firstly it would send the data to R-pi. This R-pi is connected to 4 masters in total. The protocol used for communication between R-pi and master is same as master/slave. As the system boot-up, the master starts to take input from the slave. R-pi sends an ID to Modbus, i.e. the bus where all the masters are connected. The master whose ID matches with the requested ID responds to the R-pi.

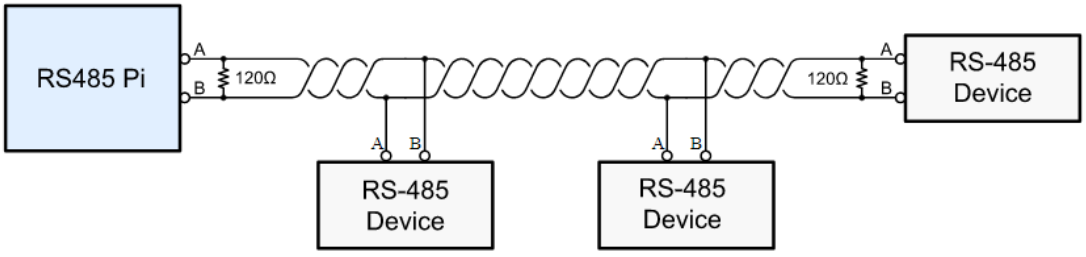

FIGURE 9.1: Communication from Master to R-pi

9.2 R-pi to Cloud Communication

9.2.1 Choice of protocol for Cloud

The protocol used for cloud communication is MQTT. The reason to use MQTT is it's lightweight and simple messaging protocol that is ideal for low bandwidth and high latency. The resource requirement of this protocol are aimed to reduce the complexity and ensure security of supply. In addition, these principles are advantageous for M2M or IoT device because of battery and bandwidth performance.

9.2.2 Working

With MQTT, data is sent from R-pi to a single destination i.e. is Cloud where the data is analyzed and interpreted. To have access to the data stored in cloud, users or other machines need to subscribe to that specific theme and can have access to data. The cloud firstly hosts and MQTT broker which is an intermediary between the subscriber and R-pi controller. This is an important distinction as machines do not directly communicate with each other but through broker. Now, R-pi send data to MQTT broker under the theme "RPM" and all subscribers, machines or users, with approved credentials and access data under this theme. Every aspect of machine's lifecycle is available for review which provides a profound opportunity to connect with this information to find defects, save cost, increase efficiency, and make planning for internet of things.

Chapter 10

Conclusion

We have made hardware prototype of our project which is working at almost 90 percent Efficiency. This hardware measure the RPM of moving spindle and give interrupt signal to our monitor when an spindle is running at much lower or faster speed or thread is broken. To give more functionality to our product we would be adding extra sensor for temperature measurement, humidity checking to monitor environment of industry as it effect the productivity of threads. More ever we would try to make our master-slave code more efficient to work flawlessly and display our data from controller to display screen. Mobile application would also be prepared to give manager access of all day action being performed in industry anytime with the help of internet connectivity.

Appendix A

Programming Codes for Software Implementation

A.1 Code for Controlling BLDC motor

```
#include <Servo.h>

Servo ESC;      // created servo object for ESC control
int potPin;     // value from the analog pin for potentiometer

void setup() {
                                    // Attach the ESC on digital pin 9
  ESC.attach(9,1000,2000);          // (pin, min pulse width, max pulse width in ms)
}
void loop() {
  potPin = analogRead(A0);                  // analog pin Ao reads the value of the potentiometer (value between 0 and 1023)
  potPin = map(potPin, 0, 1023, 0, 180);    // scale it to use it with the servo library (value between 0 and 180)
  ESC.write(potPin);                        // Sends the signal to the ESC
}

```

FIGURE A.1: BLDC Motor Code

A.2 Code for Measuring RPM of a motor

```
//defining required variables

volatile uint8_t edge = 0;  //defining 8 bit or 1 byte unsigned int
volatile unsigned int overflows = 0;
volatile unsigned long tstart = 0; //keep starting time
volatile unsigned long tdelay = 0; //Keep delay time value
volatile unsigned long tstop = 0; //keep stop time value
volatile float frequency = 0; // keep current frequency to calculate RPM
volatile float rpm = 0; // keep RMP of motor
volatile unsigned long tnow = 0;
volatile unsigned int i=0;
volatile unsigned int time_now=0;  // Current timing
int period = 50;


uint8_t ledArray[] = {4, 5, 6, 7}; // Using Pin 4,5,6 & 7 for our four sensor

float frequencyarray[4]={0,0,0,0}; // Array to keep frequency of four spindle

//one time function
```

FIGURE A.2: Defining Variables

```
//one time function

void setup(void) {
    pinMode(8, OUTPUT); // This is the analog comparator negative input.
    digitalWrite(8,LOW); // setting pin 8 as low

    // This loop will place all four pins 4,5,6 & 7 low initially
    for (byte j = 0; j < sizeof ledArray; j++)
    {
        pinMode(ledArray[j], OUTPUT);
        digitalWrite(ledArray[j], LOW);
    }

    //This "safe" code saves the current status of interrupts, turns them off (they may already be off),
    //gets the shared variable into a temporary variable, turns interrupts back on -
    //if they were on when we entered the function - and then returns the copy of the shared variable
    //_BV or Bit Value store at that specific bit of a byte
    SREG |= _BV(7);
    //Analog Comparator
    ACSR |= _BV(1) | _BV(3) | _BV(6);
    //Analog to Digital conversion
    ADCSRB |=_BV(6);

    TCCR1A = 0; // set entire TCCR1A register to 0

    TIMSK1 |= _BV(TOIE1); // enable overflow interrupt.
    TCCR1B |= _BV(CS10);// with no prescaler}
    delay(10);

    Serial.begin(9600); // For printing the frequency to the terminal

}
```

FIGURE A.3: RPM Setup Code

```
void loop(void) {

    edge = 0;
    overflows = 0;
    ACSR |= _BV(1) | _BV(3) | _BV(6);
    //ACME Analog Comparator Multiplexer Enable
    ADCSRB |=_BV(ACME);
    //ADEN
    ADCSRA &=~_BV(ADEN);

    if (i==0)
        ADMUX = B0000001;
    else if (i==1)
        ADMUX = B0000010;
    else if (i==2)
        ADMUX = B0000000;
    else if (i==3)
        ADMUX = B0000011;

    TCCR1A = 0;
    TIMSK1 |= _BV(TOIE1); // enable overflow interrupt.
    TCCR1B |= _BV(CS10);// with no prescaler}

    // Millis() gives the number of milli second since code is running
    time_now = millis();
    tdelay = millis();

    // Set the edge and frequency to 2 and 0 respectively if edge < 2 and
    // difference between total miliseconds and initial stored delay is greater than 300
    while (edge < (2)) {
        if (millis() - tdelay > 300){
        edge = 2;
        frequency = 0;
        }
    }
```

```
89
90      //Serial.println(frequency);
91
92      time_now = millis();
93      //Serial.print("RPM= ");
94      //Serial.print(i);
95      //Serial.print(",");
96
97      // if frequency is less than 300 then store frequency in freqArray, first by multiplying it to 60 to calculate RPM
98      if (frequency < 300){
99          frequencyarray[i]=frequency*60;
100     }
101     //when Freq array is filled with value then display it using below code that
102     if(i == 3){
103         // for loop display array values of RPM on Serial monitor
104         for (byte l = 0; l < sizeof ledArray; l++){
105             Serial.print( frequencyarray[l]);
106         if (l < 3)
107             Serial.print(",");
108     }
109     Serial.println();
110     }
111
112     if((frequency *60) > 5000)
113         digitalWrite(ledArray[i], HIGH);
114     else
115         digitalWrite(ledArray[i], LOW);
116
117     while (millis() < time_now + period){}
118
119 //delay(30);
120     i = (i+1)%4;
121
122 }
```

FIGURE A.4: RPM Loop Code

```
123
124 ISR(TIMER1_OVF_vect){
125     overflows += 1;
126 }
127
128 //When the interrupt is executed, the code contained in the ISR(ANALOG_COMP_vect) function is executed
129 //before the CPU returns to the main program. ISR stands for interrupt service request.
130 ISR(ANALOG_COMP_vect)
131 {
132     edge+=1;
133
134     if (edge == 1) { // Start counting edges.
135         tstart = overflows*65536 + TCNT1;
136
137     }
138     else if (edge == 2) { // Stop counting edges.
139         tstop = overflows*65536 + TCNT1;
140         frequency = ((float)16000000/(float)(tstop-tstart));
141
142         ACSR = 0;
143         TCCR1B = 0;
144     }
145 }
146
```

FIGURE A.5: RPM Interrupt Code

A.3 Master Slave Communication Code

A.3.1 Master Code

```
#include <Time.h>
#include <TimeLib.h>
#include <Crc16.h>
#include <SoftwareSerial.h>
#define LED       13    // Declare LED pin
#define MASTER_EN   6   // connected to RS485 Enable pin
Crc16 crc;
SoftwareSerial mySerial (2, 3);
SoftwareSerial forPi (4, 5);

String x;
String value;
int local,i,id=1;
float myArray[4];
char buff[10];
unsigned int x=4500;
sprintf(buff+2, x, sizeof(unsigned int));
sprintf(buff+6, y, sizeof(bool));
```

```
class Request{
    public:
    int id;
    int fc;
};
 Request rq;
void setup() {
    pinMode(LED , OUTPUT);            // Declare LED pin as output
    pinMode(MASTER_EN , OUTPUT);      // Declare Enable pin as output
    Serial.begin(9600);               // set serial communication baudrate
    digitalWrite(MASTER_EN , LOW);    // Make Enable pin low
    mySerial.begin(9600);             // Receiving mode ON
    forPi.begin(115200);
}

```

```
void loop() {

    //time_t t = now();
    //Serial.println(minute(t));
    crc.clearCrc();
    digitalWrite(MASTER_EN , HIGH);      // Make Enable pin high to send Data
    delay(5);                            // required minimum delay of 5ms
    //if (i%2 == 0)
        //mySerial.println('B');                 // Send character A serially
    //else
        rq.id = 1;
        rq.fc = 1;
        //for (i=0;i<3;i++)
        mySerial.print(id%4 + 1);
        mySerial.println('\r');
        mySerial.print(rq.fc);
        mySerial.println('\r');
    id++;
    mySerial.flush();                      // wait for transmission of data

    digitalWrite(MASTER_EN , LOW);       // Receiving mode ON

    // x = mySerial.readString();

    for (i=0;i<4;i++)
        myArray[i] = mySerial.parseFloat();

    for (i=0;i<4;i++){
        Serial.print(myArray[i]);
        Serial.print(",");
    }
    Serial.println();
    //Serial.println(x);
    //mySerial.read();
    //value = mySerial.readStringUntil('\n');
    //crc.updateCrc(x.toInt());
    //local = crc.getCrc();
    //if (local == value.toInt())
        //Serial.println(x);

    delay(500);
}
```

A.3.2 Slave Code

```
#include <Crc16.h>
//A cyclic redundancy check is an error-detecting code
//commonly used to detect accidental changes to raw data.
//Blocks of data entering these systems get a short check value attached,
//based on the remainder of a polynomial division of their contents
#define LED       13
#define SLAVE_EN  6
Crc16 crc;
//Software serial is used to allow serial communication on digital pin
#include <SoftwareSerial.h>
SoftwareSerial mySerial (2, 3); // defining Serial pins
float rpm_rec = 123.0; //set demo value of RPM
int rec_id,rec_fc;
//int id = 1,time_check;
String value;
// setting 4 slave each having 4 RPMs
float rpm1[4] = {20.3,10.6,9.8,59.2},rpm2[4] = {41.2,69.4,42.3,88.1},
      rpm3[4] = {33.3,11.1,0,44.4},  rpm4[4] = {102.3,45.9,102.0,63.6};
float *rpm;
int i;

volatile uint8_t edge = 0;  //uint8_t
volatile unsigned int overflows = 0;
volatile unsigned long tstart = 0;
volatile unsigned long tstop = 0;
volatile float frequency = 0;
//volatile float rpm = 0;
volatile unsigned long tnow = 0;

```

```
void setup() {
    pinMode(7, INPUT); // This is the analog comparator negative input.
    //This "safe" code saves the current status of interrupts, turns them off (they may already be off),
    //gets the shared variable into a temporary variable, turns interrupts back on -
    //if they were on when we entered the function - and then returns the copy of the shared variable
    //_BV or Bit Value store at that specific bit of a byte
    SREG |= _BV(7);
    ACSR |= _BV(1) | _BV(3) | _BV(6);
    ADCSRB |=_BV(6);
    TCCR1A = 0;
    TIMSK1 |= _BV(TOIE1); // enable overflow interrupt.
    TCCR1B |= _BV(CS10);// with no prescaler}
    delay(10);

    pinMode(LED , OUTPUT);                      // Declare LED pin as output
    pinMode(SLAVE_EN , OUTPUT);                 // Declare Enable pin as output
    Serial.begin(9600);                         // set serial communication baudrate
    digitalWrite(SLAVE_EN , LOW);               // Make Enable pin low
    mySerial.begin(9600);                                       // Receiving mode ON
}
```

```
void loop() {

    //parseInt() returns the first valid (long) integer number from the serial buffer.
    rec_id = mySerial.parseInt();
    rec_fc = mySerial.parseInt();
    //rpm_rec = RPM();

    decision(rec_id);
}

void decision (int id){

    if(id == 1)                         // if available data is A
        rpm = rpm1;
    else if (id == 2)
        rpm = rpm2;
    else if (id == 3)
        rpm = rpm3;
    else if (id == 4)
        rpm = rpm4;

    crc.clearCrc();                     //clearing CRC
    crc.updateCrc(rpm_rec);             //updating for recoreded rpm
    value = crc.getCrc();               //get CRC
    //time_check =
    digitalWrite(SLAVE_EN , HIGH);          // Make Enable pin high to send Data
    delay(5);                               // required minimum delay of 5ms

    for (i=0;i<4;i++){
        mySerial.print(rpm[i]);             // Send character A serially
        mySerial.println('\r');
    }
    //mySerial.print(',');
    //mySerial.println(value);

    for (i=0;i<4;i++){
        Serial.print(rpm[i]);
    }
        Serial.println();
        mySerial.flush();                   // wait for transmission of data
        delay(5);
        digitalWrite(SLAVE_EN , LOW);       // Receiving mode ON
}
```

A.4 Master to R-pi Communication Code

```
import time
import serial
import RPi.GPIO as GPIO
from time import sleep
```

```
# GPIO.BOARD option specifies that you are referring to the pins by the number of pin in the board
GPIO.setmode(GPIO.BOARD)

# The GPIO pin number 7 on the Raspberry Pi is made HIGH because the pin 7 of Pi is
# connected to DE & RE of RS-485.It is made HIGH because it makes RPi to send values to RS-485
GPIO.setup(7, GPIO.OUT, initial=GPIO.HIGH)

# Initiate the serial class at the pins GPIO14 & GPIO 15 (Serial0 Port) with various
# information like which serial port, baud rate, parity and stop bits.
send = serial.Serial(
    port='/dev/serial0',
    baudrate = 9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)

# The variable 'rpm_array' with array of angle values is defined, these values will be sent via serial communication.
rpm_array = [0,10,45,90,135,180,135,90,45,10,0]
```

```
# The function send.write(str(x)) sends the values to serial port to the RS-485 one by one written
# inside the while loop as it executes continuously. The values are sent with a delay of 1.5 seconds.
while True:
 for x in i:
     send.write(str(x))
     print(x)
     time.sleep(1.5)
```

FIGURE A.6: Code for Master to R-pi Communication

References

[1] A2212 brushless motor pinout, wiring, features, specs amp; datasheet. URL https://components101.com/motors/2212-brushless-motor.

[2] Lammert Bies © 1997-2019. Rs485, specifications and in depth tutorial. URL https://www.lammertbies.nl/comm/info/rs-485.

[3] Qingguang Chen, Lingyun Xue, and Huanle Rao. Rotational speed measurement of ring spinning based on magnetic sensor. *Measurement Science and Technology*, 28(11):115106, 2017.

[4] M Hossain, A Abdkader, A Nocke, R Unger, F Krzywinski, MMB Hasan, and Chokri Cherif. Measurement methods of dynamic yarn tension in a ring spinning process. *Fibres & Textiles in Eastern Europe*, 2016.

[5] Jack. Esc 30a datasheet - diy quadcopter, Aug 2019. URL http://www.rcdronegood.com/esc-30a-datasheet/.

[6] RZtronics and Instructables. Control brushless motor using arduino, Sep 2017. URL https://www.instructables.com/Control-Brushless-Motor-Using-Arduino/.