CHAPTER-4

METHODOLOGY

The chapter describes the methodology used for system development. First the reading and testing of flex sensor is done and controller is programmed accordingly. To ensure the optimize solution to the problem statement for current thesis two approaches are considered. First is implementation of designed system with the selected components and check the performance with experimental set up. Testing is performed on age group of 18-25 years during different time spam of year, to check the variation in the output of sensor due to change in weather conditions and system is found stable w.r.t to change in weather in Dehradun, Uttrakhand, India where temperature variation is from 4^0C- 40^0C during winters and summer season.

The complete system is analyzed with the help of specially designed LabVIEW GUI. Two different GUI are designed one for analyzing the flex sensor output values at helmet node and other for the complete system analysis for ignition of vehicle which depends on flex sensors average value and RFID code.
Second approach is modelling of microcontroller chip for the proposed system with required features and I/O pins to provide an optimize solution in form of specially designed microcontroller. Two high speed microcontroller chips are designed one for the helmet node and other for two-wheeler node.

By applying hypothesis on the statistical data collected from different samples threshold value of flex sensors is calculated for setting the appropriate level to ignite the vehicle.

4.1 Read the Flex Sensor

Flex sensor is used for sensing change strain exerted between head and helmet, when driver wears it. Flex is a low cost sensor and it is simple to use it. It is basically a variable resistor, value of which varies according to change in strain. The value of resistance decreases with increase in change in angle of the sensor. Fig.4.1 shows the arrangement for reading the flex sensor.

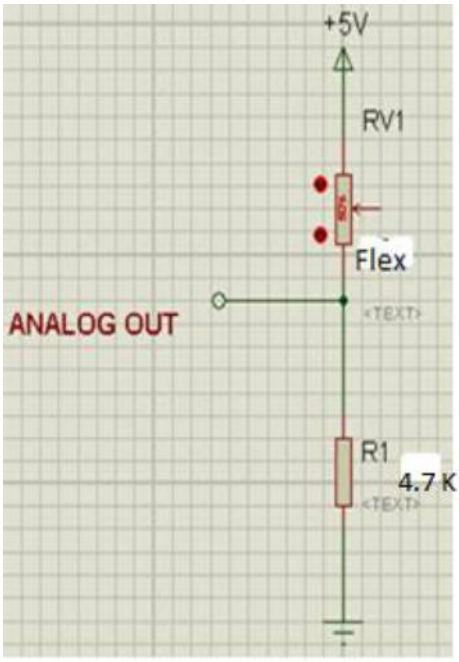

Fig.4.1 Arrangement for reading flex sensor

Output voltage for flex is given as equation below-

$$flex(V) = flexADC * \left(\frac{Vcc}{1023}\right) \quad \text{Eq.4.1}$$

$$flex(V) = \left(\frac{flexR}{R+flexR}\right) * Vcc \quad \text{Eq.4.2}$$

$$flex(R) = R * \left(\frac{Vcc}{flex(V)-1}\right) \quad \text{Eq.4.3}$$

R = 4.7 K

Here-

R = Resistance connected to flex

Flex(R)= Resistance by Flex sensor

Vcc = Input Voltage (V)

V = Analog output

4.1.1 Placement of the Flex Sensors

Helmet standards agencies identified four pressure points in the helmet where maximum change in pressure exerts when an accident happens. As shown in Fig.4.2 these points are B (front side of head), Z&X (right and left side of head slightly above to ears) ,R (back side of head) [law.resource.org].

While wearing helmet three points B, Z, X exerts pressure, so these three points are considered for placement of flex sensors, so that it can be verified that driver is wearing helmet or not. While designing microcontroller chip four analog sensor input pins are considered so that the same microcontroller can be used in future to analyze accident impact on head.

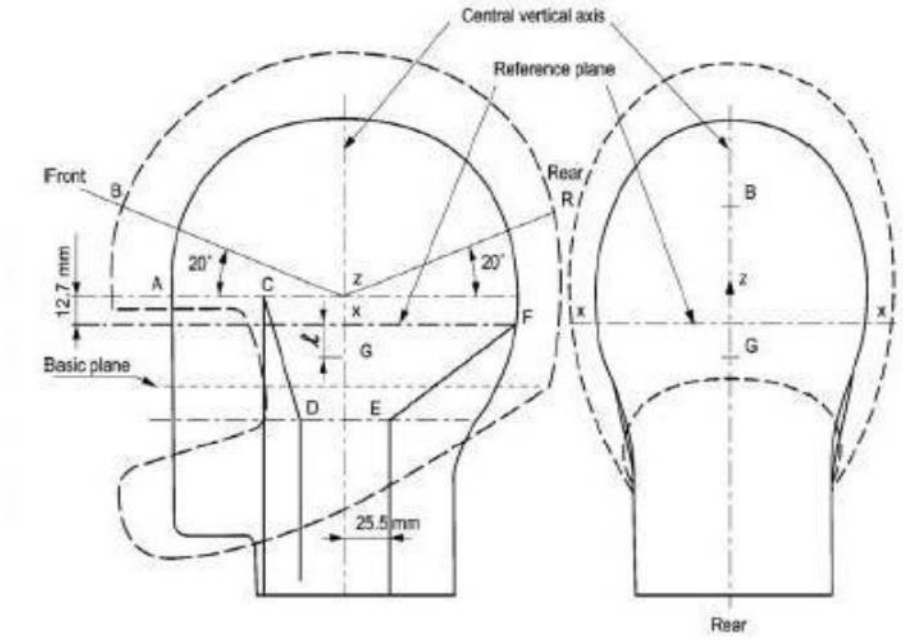

Fig.4.2 Placement of flex sensors

4.2 RFID Code Extraction

To identify the authorized person, RFID tags are used with RFID reader which is placed at two wheeler node. Each RFID tag has a unique twelve byte code which needs to be extracted from the tag and same needs to write in the program. Then controller has to identify predefined code when user swipes RFID

tag on the RFID reader. A small circuit needs to be developed for extracting the code from RFID tag. The extracted code can be checked on software terminal V1.9. Fig.4.3 shows the circuit diagram for code extraction of RFID tag with Atmega microcontroller.

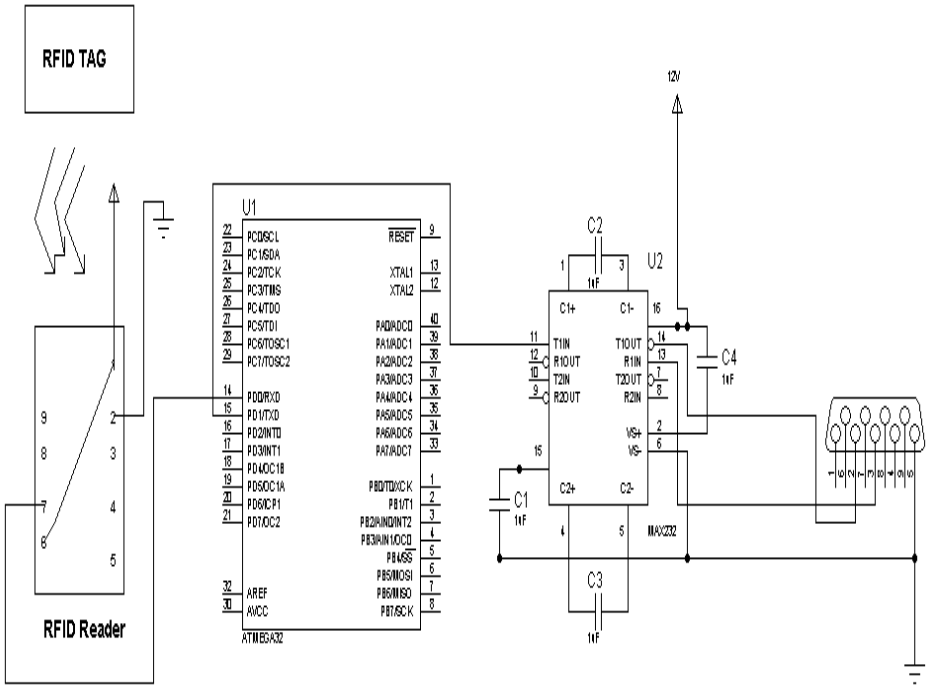

Fig.4.3 Circuit diagram for RFID code extraction

Circuit Diagram

Connections:

1. Connect 2 and 3 no. pins of DB9 connector to 14(R2-IN) and 13(T2-OUT) Pins of MAX232 IC respectively.

2. Connect 11(T2-IN) pin of MAX232 to 15(TX) of Atmega32.

3. Connect capacitor between 1 and 3 of MAX232 where (+) terminal of capacitor is connected to 1 no. pin and 3 with (–) terminal of capacitor.

4. Connect capacitor between 4 and 5 of MAX232 where (+) terminal of capacitor is connected to 4 no. pin and 5 with (–) terminal of capacitor.

5. Connect capacitor between 2 and 16 of MAX232 where (+) terminal of capacitor is connected to 2 no. pin and 16 with (–) terminal of capacitor.

6. Connect capacitor between 6 and 15 of MAX232 where (+) terminal of capacitor is connected to 6 no. pin and 15 with (–) terminal of capacitor.
7. Connect 7 no. pin (data) of RFID to 14(RX) no. of pin of Atmega32.
8. Connect 1 and 6 pin of RFID to Vcc(+5V) and 2 no. pin to Ground.

DB9 connector directly can be connected with PC/laptop to see the code on Terminal V1.9, as shown in Fig.4.4.

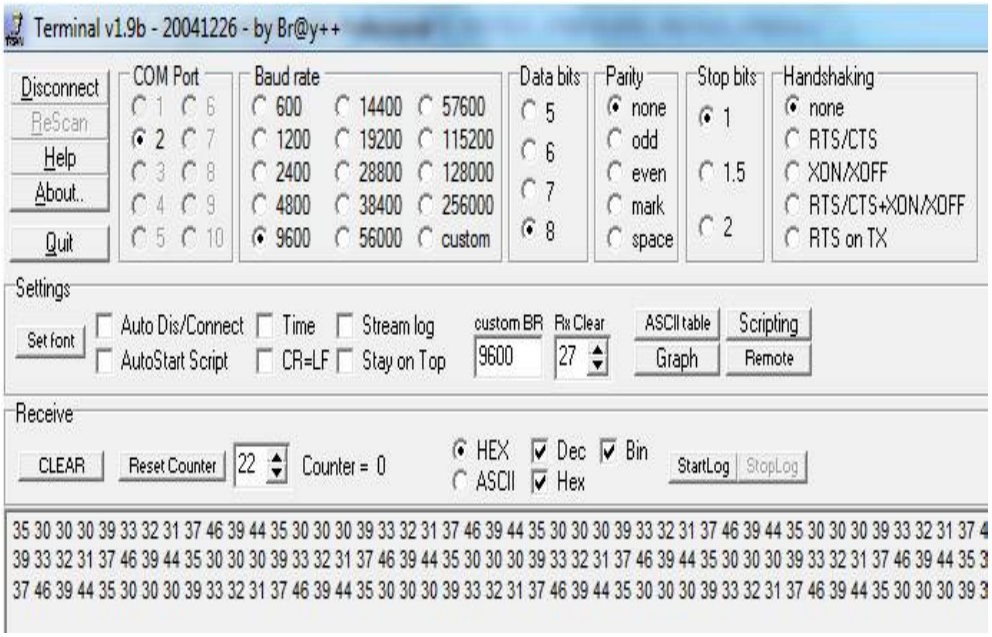

Fig.4.4 Terminal v1.9 for data process from RFID tag

At the two-wheeler node AND operation is performed on the RFID value and average value of flex sensors, if it matches to pre-defined values then only vehicle will be ignited.

4.3 Program Functions for the System

Program functions are written for each section to develop the system.

4.3.1 Program Functions for the Helmet Node

The following are the functions written for the helmet node.

>>>setup() for HELMET NODE

```
void setup()
{
  lcd.begin(20,4); // initialize 20*4 LCD in helmet node
  Serial.begin(9600);// initialize serial communication in helmet node

 }
```

>>>loop() for HELMET NODE

```
void loop()
{
  READ_FLEX_sensor_HN();
  PRINT_FLEX_sensor_HN_LCD();
  Serial.print(FLEX_AVERAGE);
}
```

Function to read flex sensors

```
READ_FLEX_sensor_HN()
{
  Flex1_Read_BYTE = analogRead(Flex_Pin_1=A0);// read flex sensor connected at A0
  Flex2_Read_BYTE = analogRead(Flex_Pin_1=A1); // read flex sensor connected at A1
  Flex3_Read_BYTE = analogRead(Flex_Pin_1=A2); // read flex sensor connected at A2
```

```
 int flex1_LEVEL= Flex1_Read_BYTE ;// scale the voltage level of flex sensor1
 int flex2_LEVEL = Flex2_Read_BYTE; // scale the voltage level of flex sensor 2
 int flex3_LEVEL= Flex3_Read_BYTE; // scale the voltage level of flex sensor 3
 int FLEX_AVERAGE=((flex1_LEVEL+flex2_LEVEL+flex3_LEVEL)/3);// average the flex sensor levels
}
```

Function to print on LCD

```
PRINT_FLEX_sensor_HN_LCD()
{
lcd.setCursor(0,0);
lcd.print(flex1_LEVEL);
lcd.setCursor(0,1);
lcd.print(flex2_LEVEL);
lcd.setCursor(0,2);
lcd.print(flex3_LEVEL);
lcd.setCursor(0,3);
lcd.print(FLEX_AVERAGE);
}
```

4.3.2 Program Functions for the Two-Wheeler Node

The following are the functions written for the two-wheeler node.

>>>setup() for two wheeler Node

```
#include <LiquidCrystal.h>//header of LCD
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);// connect pins of Arduino to RS,E,D4,D5,D6,D7 of LCD
#include<SoftwareSerial.h>// header of soft serial library
SoftwareSerial mySerial(9,10);//make pin 9 and 10 pins of Arduino as Rx and Tx
```

```
#define IGNITION_RELAY 7 //assign pin 7 to IGNITION_RELAY

void setup()
{
lcd.begin(20,4);// initialize 20*4 LCD
mySerial.begin(9600);// initialize the serial communication using Soft serial library-9600-8-N-1
Serial.begin(9600);// initialize the serial communication-9600-8-N-1
inputString_serial_TWN.reserve(100);// reserve the 100 byte for inputString_serial_TWN
pinMode(IGNITION_RELAY, OUTPUT);// set pin 7 as output
}
```

>>>loop() for Two -wheeler Node

```
void loop()
{
Receive_Data_RFID_TWN();
Check_Data_RFID_TWN();
Access_Check_RFID_TWN();
Serial_Event_serial_TWN();

if (stringComplete_serial_TWN)
  {
    lcd.setCursor(0,2)// set LCD cursor at column0 and Row2
    lcd.print(inputString_serial_TWN);//print inputString_serial_TWN on LCD
    Serial.println(inputString_serial_TWN);// send inputString_serial_TWN seial

    if(inputString_serial_TWN [0]>='2')&&( inputString_serial_TWN [1]>='1')&&(
```

```
                inputString_serial_TWN [1]>='2'))
    {
    lcd.clear();// clear the previous contents from LCD
    lcd.setCursor(0,3); //set LCD cursor at column0 and Row3
    lcd.print("IGNITION VERIFIED ");//print string on LCD
    digitalWrite(IGNITION_RELAY, HIGH); // make pin IGNITION_RELAY
HIGH
     }
   else
   {
    lcd.clear();//clear the previous contents from LCD
    lcd.setCursor(0,3);// set LCD cursor at colun0 and Row3
    lcd.print("IGNITION NOT VERIFIED "); print string on LCD
    digitalWrite(IGNITION_RELAY,LOW); // make pin IGNITION_RELAY
HIGH
     }
  inputString_serial_TWN = "";// clear the previous contents from string
inputString_serial_TWN
  stringComplete_serial_TWN = false;
   }
}
```

>>>Function to store the serial byte in defined string of two-wheeler node

SerialEvent occurs when a new data received at the hardware serial RX. This routine is checked between each time loop() runs, so by using delay inside loop response multiple bytes of data can be checked.

```
void serial_Event_TWN()
{
        while (Serial.available()>0)
```

```
        {
         Char serial_ BYTE= (char)Serial.read();  // record new byte
          inputString_serial_TWN+= serial_ BYTE;      // add new byte with the inputString_serial_TWN
                 if (serial_ BYTE == 0x0D)  // if the incoming character is an enter, then set a flag
                {
                stringComplete_serial _TWN= true;// Boolean logic if True
                 }
        }
}
```

>>> **Define RFID TAG twelve byte code to compare with received serial TAG data**

```
char RFID_TWN _Saved_Tags [3][12]={
                                    {'5','0','0','0','9','2','B','E','9','3','E','F'},//12 Byte from RFID TAG1
                                    {'5','0','0','0','9','2','E','A','2','C','0','4'},      //12 Byte from RFID TAG2
                                    {'5','0','0','0','9','3','2','1','7','F','9','D'}       //12 Byte from RFID TAG3
                                   };
```

>>> **Function to receive 12 byte from RFID**

```
void Receive_Data_RFID_TWN()
{
  if(mySerial.available()>0)
    {
     Char RFID_serial_byte=mySerial.read();// read serial data
     RFID_string_TWN[total_count_byte]= RFID_serial_byte;// store data in string
```

```
    total_count_byte ++;// increment the counts
   } }
```

>>> **Function the check data from RFID**

```
void Check_ Data_  RFID_ TWN()
  {
  if ( Total_ Count_ byte ==12)
     {
      entry_ control_ RFID_TWN= TRUE;
      for (k=0; k<3; k++)
     {
      for (j=0; j<12; j++)
        {
        If (RFID_ TWN_ Saved_ Tags[k][j]== RFID_String _TWN[j])
          {
           Tag_ Check_RFID_TWN= TRUE;
          }
        else
          {
           Tag_ Check_RFID_TWN= FALSE;
           break;
          }
      }
  If (Tag_ Check _RFID_TWN== TRUE)
      {
    Tag_ Status _RFID_TWIN=TRUE;
     }
    }
    RFID_ String _TWN =0;
         }
```

```
  }
```

>>> Function to authenticate the data from RFID

```
void Access_ Check_RFID_TWN()
{
  if(Entry_ Control_RFID_TWN== TRUE)
  {
    if(Tag_ Status_RFID_TWN== TRUE)
    {
       lcd.set Cursor(0,1); set LCD cursor at column0 and Row1
       lcd.print ("ACCESS GRANTED FOR IGINTION");  // print string on LCD


    }
 else
    {
    lcd.set Cursor(0,1); set LCD cursor at column0 and Row1
    lcd.print ("ACCESS DENIED FOR IGINTION");// print string on LCD
    lcd.setCursor (0,3); set LCD cursor at colun0 and Row3
    lcd.print ("NOT VERIFIED");// print string on LCD
      }
  entry_control_RFID_TWN=FALSE;
  Tag_ Status_RFID_TWN=  FALSE;
 }
}
```

4.4 LabVIEW GUI

An intelligent helmet is designed for two-wheeler. The analysis process of the system is done with the help of LabVIEW GUI. LabVIEW is used for analysis of real time hardware interfacing with system by using virtual components.

Components used to design LabVIEW GUI

Visa Configure Serial port- This port is used initialize the serial port specified by VISA resource name with required settings. Fig.4.5 shows the visa configure serial port.

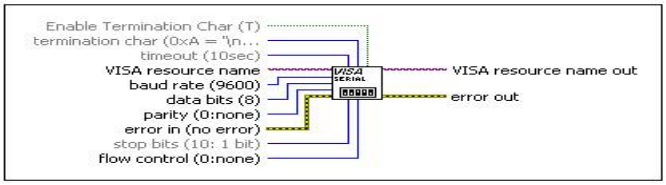

Fig.4.5 Visa Configure Serial port

VISA resource name (COM number)

Right click on it and select **create** then select **control** to choose appropriate COM port.

Baud rate- '9600'

Data bits- '8 bits'

Parity- 'none'

Stop Bit- ' 1'

Flow control – 'none'

Visa resource name out- Connect this pin to **Visa resource name** of **VISA serial read** block.

Error out-connect this pin to error pin of **VISA serial read**

VISA serial Read

It reads the identified number of bytes from the device or interface identified by **VISA resource name** and sends the data in **read buffer**. It reads the data available at serial port from the device linked.

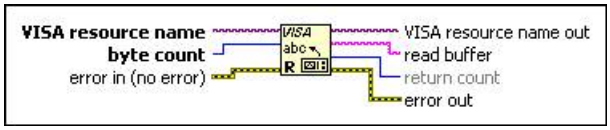

Fig.4.6 VISA serial Read

Byte Count- Right click on it and select **create** to the **indicator** to count the byte at serial port.

Read Buffer - Right click on it and select **create** to the **indicator** to check the string value at serial port.

Visa resource name out-Connect this pin to **Visa resource name** of **VISA close** block

Error out- Connect this pin to error pin of **VISA close**.

Error in- connect this pin to error out pin of **VISA configure serial port**.

Match Pattern

It searches for expression in string beginning at offset, and when it get the string it matches the string with predefined data.

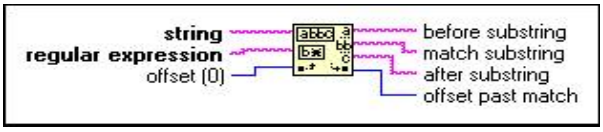

Fig.4.7 Match Pattern

String-Connect this pin to read buffer pin of **VISA read.**

Regular expression- Right click on it and select **create** to the **constant**.

After substring-connect this pin to **string** of **Decimal String to Number** block and **create** to **constant.**

Decimal String to Number- It converts the numeric characters in to the **string**, start at **offset**, to a decimal integer and return it in **number.**

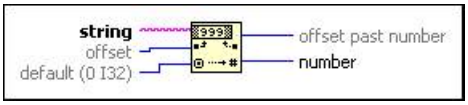

Fig.4.8 Decimal String to Number converter

Number- Connect this pin to the input of **waveform chart.**

VISA Close - Closes a device session or event object specified by VISA resource name

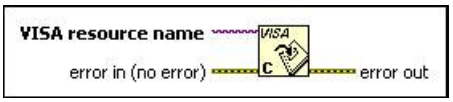

Fig.4.9 VISA Close

VISA Serial Write- It is used to write the data from the write buffer to the device or interface identified by VISA resource name

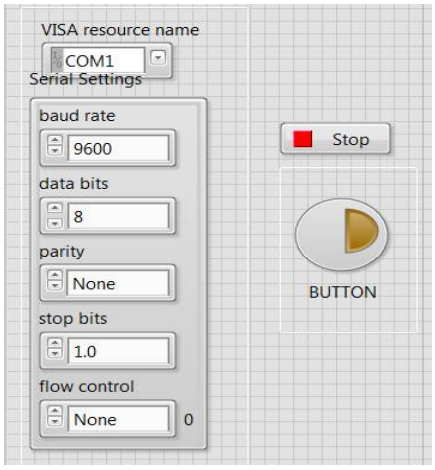

Fig.4.10 Defining VISA resource name

Visa Configure Serial port- It sets the serial port identified by VISA resource name to the specified settings.

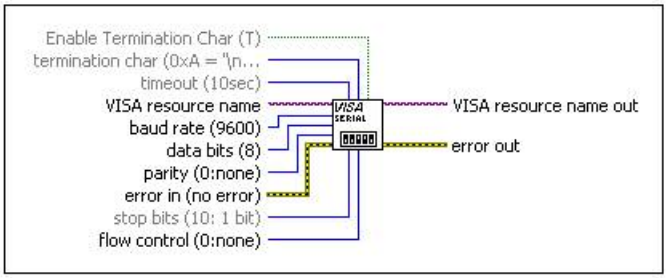

Fig.4.11 Visa Configure Serial port

VISA resource name (COM number)- Right click on it select **create** and then select **control** to choose appropriate COM port

Baud rate- '9600'

Data bits- '8 bits'

Parity- 'none'

Stop Bit- '1'

Flow control – 'none'

Visa resource name out- Connect this pin to **Visa resource name** of **VISA serial read** block.

Error out-connect this pin to error pin of **VISA serial read**

VISA Serial Write- It writes the data from write buffer to the device or interface stated by VISA resource name.

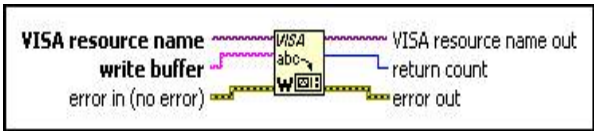

Fig.4.12 VISA Serial Write

VISA resource name- Connect this pin to **VISA resource name out** pin of **VISA configure serial port**.

Write Buffer - It comprises of the data to be written to the device.Right click on it and select **create** to the **constant** to send string at serial port.

Visa resource name out-Connect this pin to **Visa resource name** of **VISA close** block

Error out- Connect this pin to error pin of **VISA close**.

Error in- connect this pin to error out pin of **VISA configure serial port**.

VISA Close – It Close a device session or event object specified by VISA resource name.

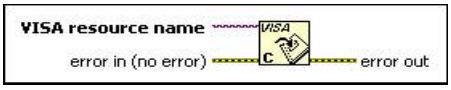

Fig.4.13 VISA Close

4.5 LabVIEW Interfacing with Proteus Simulation Software

Before interfacing with actual hardware LabVIEW GUI can also be checked by interfacing to Proteus Model Virtual Serial Port. VSPE needs to be installed which creates the virtual bridge between the serial port with the LabVIEW simulator. The real time data from the hardware or Proteus Model data is taken through serial port of the system.

The system is first designed on Proteus simulation model and checked the feasibility and interface Proteus model with LabVIEW VSPE software is used.

Virtual Serial Port Emulator (VSPE)

To set up the interfacing install VSPE software and open the window. Click on device and create as shown in Fig.4.14. A pop will appear and select 'pair' as device type and click next, as shown in Fig.4.15. Then assign the COM of COMPIM of Proteus and COM in LABVIEW as shown in Fig.4.16. Then click add button on software it will paired and show the status in device manager. Fig.4.17 shows the COMPIM configuration in Proteus to interface with LABVIEW. Fig.4.18 shows the COMPIM and Arduino connection in Proteus.

The COMPIM model is a Physical Interface Model (PIM) of a serial port. It receives serial data in buffer and assumed to the circuit as input signal. The data from the CPU or UART model appears at the PC's physical COM port.

Fig. 4.14 shows the VSPE window which opens on clicking the software icon.

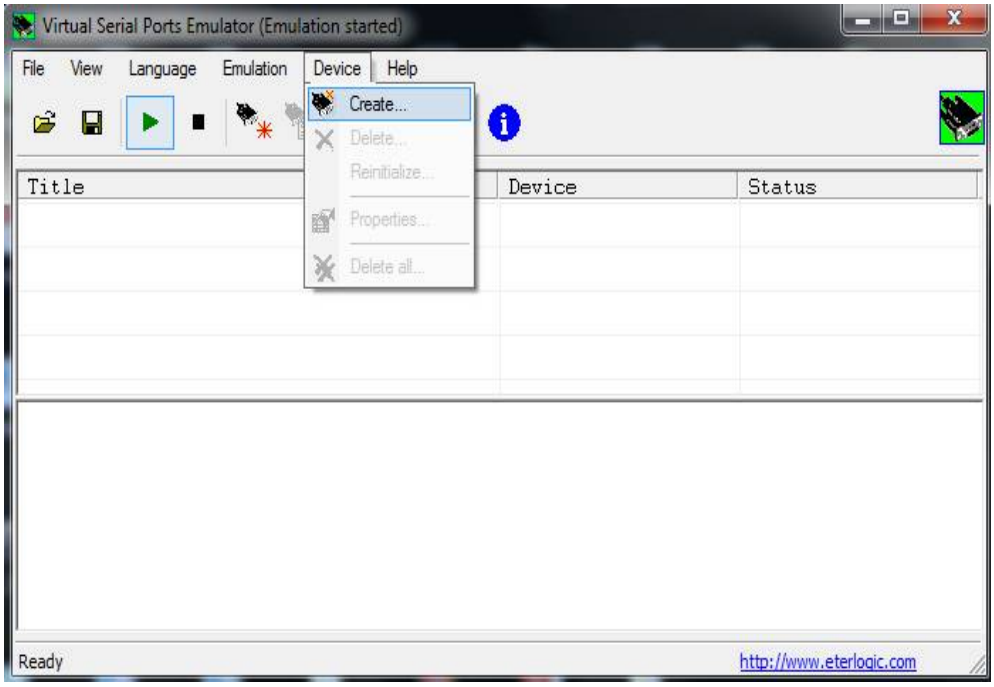

Fig.4.14 VSPE window

Fig. 4.15 shows the VSPE window for pairing the two virtual ports.

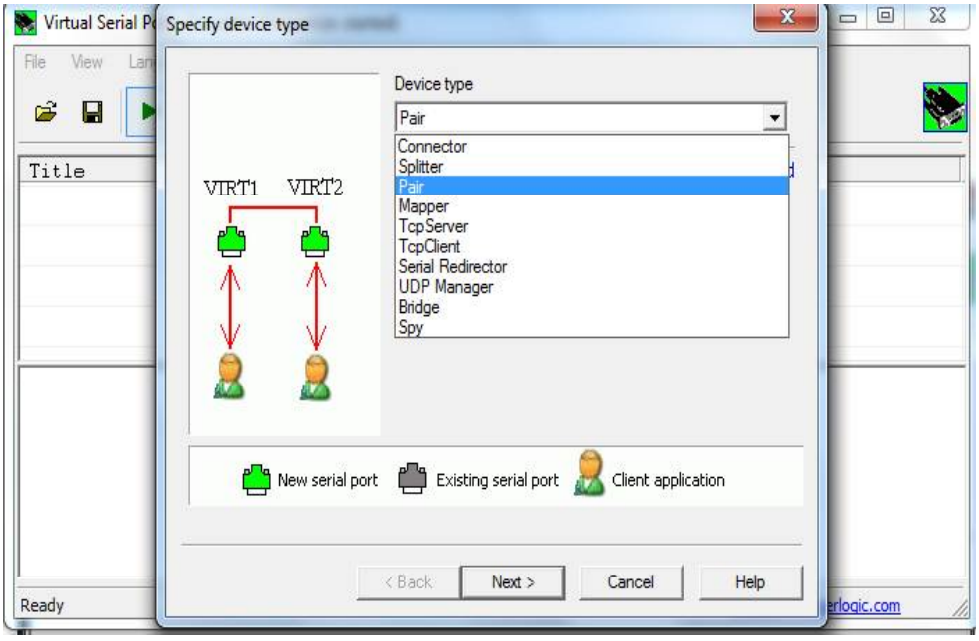

Fig.4.15 VSPE window for pairing two virtual ports

Fig. 4.16 shows how to assign COM ports and then ‘Finish’ the pairing process.

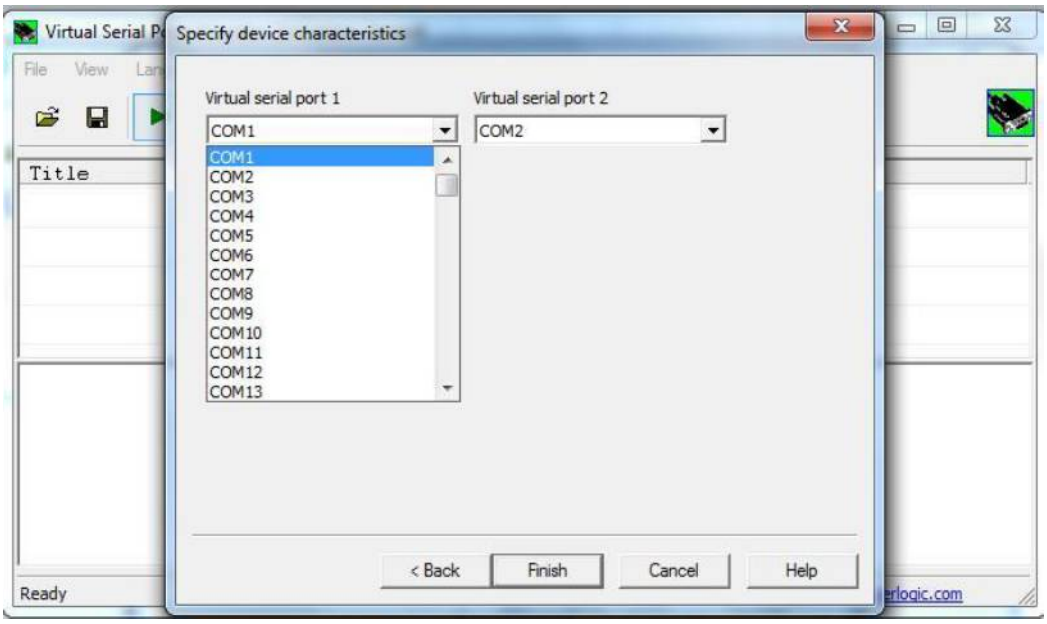

Fig.4.16 VSPE window for assigning COM port number

Fig.4.17 shows the paired COM ports in VSPE. Let’s assume COM1 is with Proteus model and COM2 is with LabVIEW.

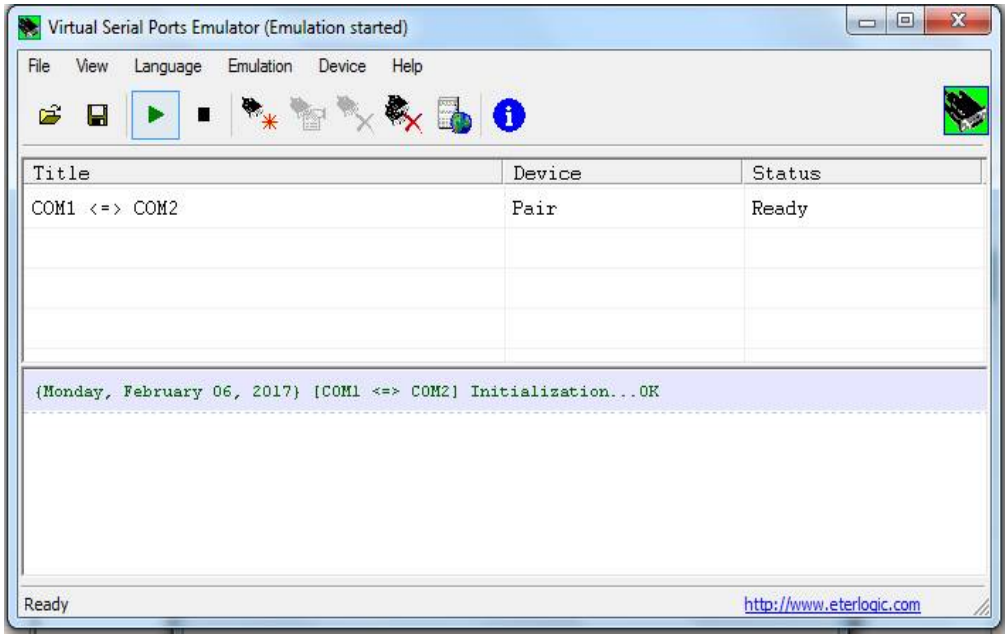

Fig.4.17 VSPE window showing paired COM ports

Fig.4.18 shows how to interface COMPIM with controller in Proteus model.

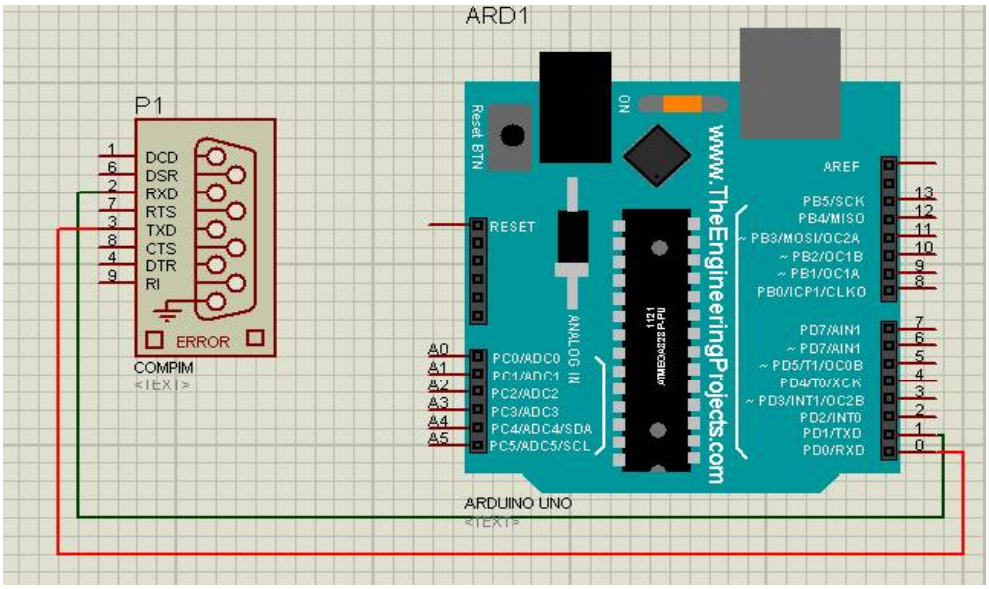

Fig.4.18 COMPIM and Arduino connection in Proteus

Fig.4.19 shows how to configure the COMPIM in Proteus model. The edit component POP up appears on right click on the COM port in the model.

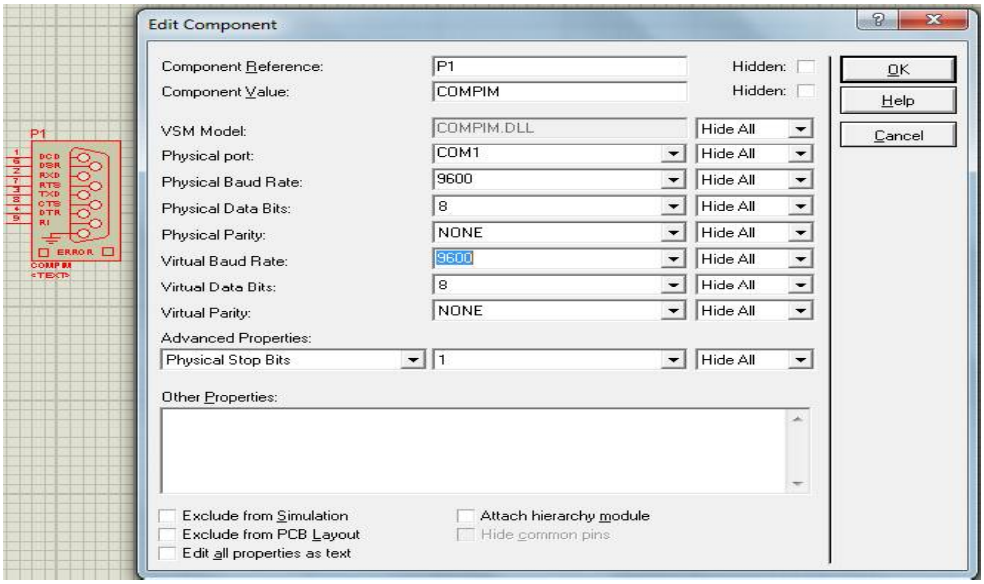

Fig.4.19 Configure the COMPIM in Proteus to interface with LabVIEW

Fig.4.20 shows the Proteus Simulation Model displaying the sensor value on virtual terminal after COM port pairing with VSPE.

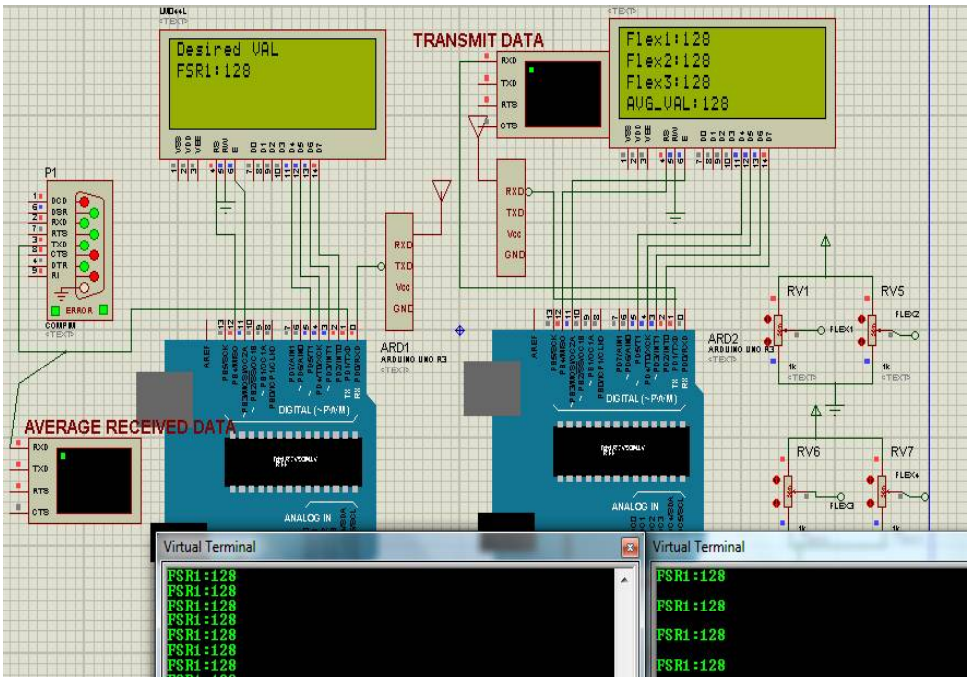

Fig.4.20 Proteus Simulation Model showing sensor value at virtual terminal

4.5.1 Program Functions for LabVIEW GUI

The following functions are written for LabVIEW GUI.

>>> **setup() for data logger**

```
void setup()
{
  lcd.begin(20,4);  // initialize LCD in data logger
  Serial.begin(9600);  // initialize serial communication as 9600-8-N-1
  Input String_ serial_datalogger.reserve(100); // reserve 100 bytes for the inputString_serial_datalogger
}
```

>>>loop() for data logger

```
void loop()
```

```
{
 if (stringComplete_serial_datalogger)  // print the string when a 0x0D(enter) comes in inputString_serial
 {
  lcd. setCursor(0,0);// set the cursor at column 0 and row 0
  lcd.print (inputString_serial_datalogger);//print an inputString_serial data on LCD
  Serial.print("STRING_FOR_LABVIEW:");//send serial label for LABVIEW
  Serial.println(inputString_serial_ datalogger);// print an inputString_serial data serially

   inputString _serial_datalogger= "";// make inputString _serial string blank again
  stringComplete_serial _datalogger= false;// Boolean logic if false
 }
}
```

>>>>>>**Function to store the serial byte in defined string of data logger (SERVER)**

The serial event occurs when a new data is received from the hardware serial RX. This routine is run between each time loop() runs.

```
void serial_Event_datalogger()
{
        while (Serial.available()>0)
         {
          Char serial_ BYTE= (char)Serial.read();  // record new byte
 inputString_serial_datalogger += serial_ BYTE;      // add new byte with the inputString_serial_
//datalogger
```

```
                if (serial_ BYTE == 0x0D)  // if the incoming character is an
enter, then set a flag
                {
                stringComplete_serial _datalogger= true;// Boolean logic if True
                }
        }
}
```

4.6 Programming Flow Chart for the System Development

Fig.4.21 shows the flow chart for all programming steps to program the helmet node.

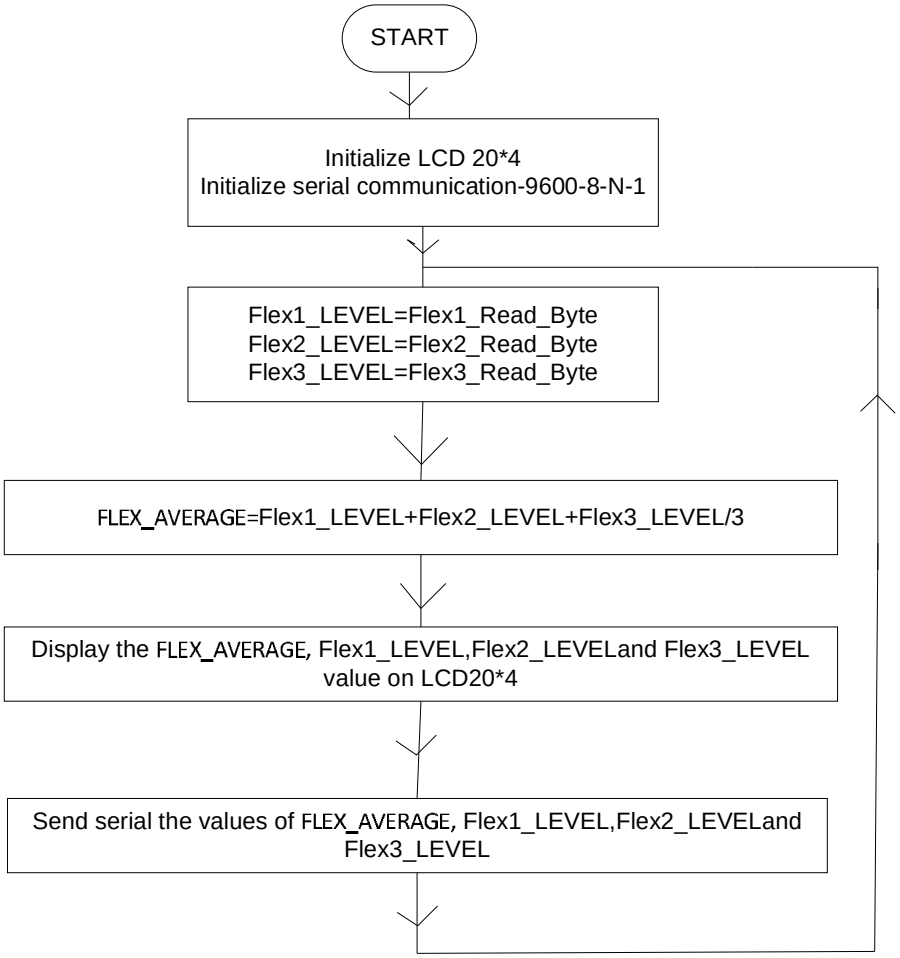

Fig.4.21 Flow chart for helmet node

The functions for LCD and serial communication are initialized. After the initialization the system will wait for the signals from three flex sensors placed in helmet. Average value of the three sensor outputs is taken, which is displayed on LCD for experiment purpose and sent to the two wheeler node through serial communication at baud rate 9600 bps. To send data on cloud server node MCU is connected to Arduino UNO and receive the data on www.thingspeak.com .

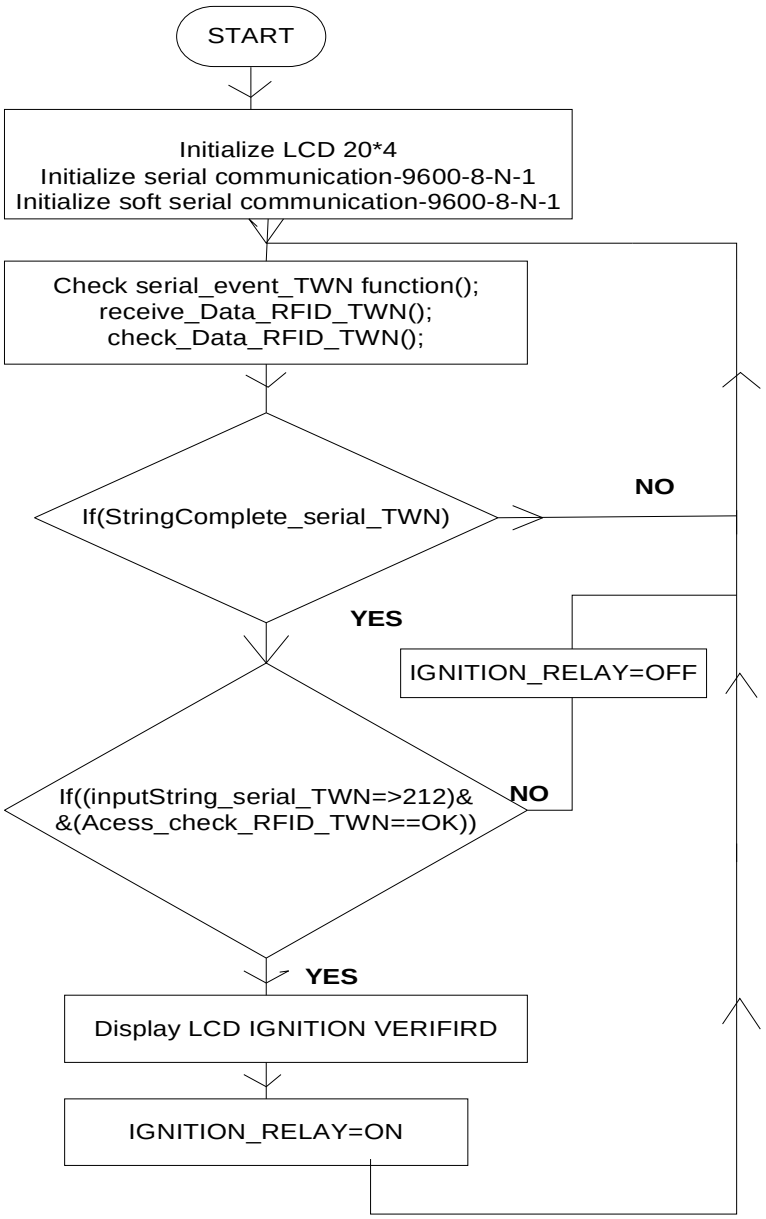

Fig.4.22 Flow chart for two-wheeler node

Fig.4.22 shows the flow chart for two-wheeler node. The functions for LCD and serial communication are initialized. After the initialization the system will receive RFID data when card is swiped on RFID reader and checks it by comparing the received data with pre-defined code in the program. If it matches with predefined code then a check is performed on average value of flex sensors received from helmet node and RFID tag code, if both matches with predefined values then vehicle will be ignited otherwise not.

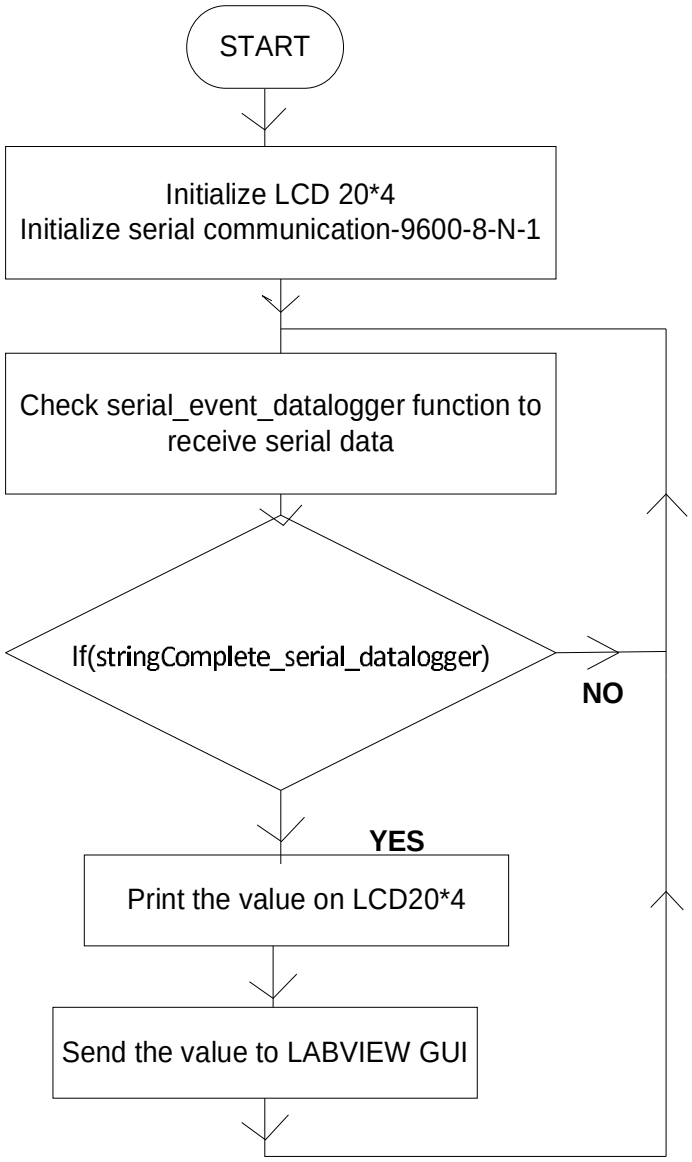

Fig.4.23 Flow chart for LabVIEW GUI

Fig.4.23 shows the flow chart for LabVIEW GUI program. The functions for LCD and serial communication are initialized. Then a check is performed for serial event to receive data, if a complete string is received then it will be printed on LCD and sent to the LabVIEW GUI otherwise the system will again check for valid serial data.

4.7 Experimental Research

Experimental research for the system is done to ensure the selection of appropriate experimental design. It involves following steps-

- Define and state the problem
- Develop a hypothesis
- Design and conduct experiments to test the hypothesis
- Collect data
- Analyze the data
- Interpret the data
- Conclude about the hypothesis

Concept of hypothesis

A hypothesis is a proportion, a tentative assumption which a researcher wants to test for its logical or empirical consequences. In problem oriented research, it is necessary to formulate a hypothesis. In such researcher hypothesis are generally concerned with the causes of a certain phenomenon or a relationship between two or more variables under investigation.

Hypothesis testing

Steps involved in testing a hypothesis-

- Formulate a ‘hypothesis’
- Set up a suitable significance level
- Choose a ‘test criterion’
- Compute the statistic from the samples

- Make the decision

If a hypothesis is of the type $\mu = \mu_{H0}$, then it is called specific hypothesis but if it is of the type $\mu > \mu_{H0,}$ $\mu < \mu_{H0}$ then it is called composite or nonspecific hypothesis.

If results do not support null hypothesis which means something else is true, then it is known as alternative hypothesis.

't'-test

't'- test can be performed on the samples with less than thirty samples of same type. For that null hypothesis is to be defined first and following the formulas for T-test conclusion can be made.

$$t = \frac{\bar{x} - \mu_{H0}}{\sigma_s / \sqrt{n}} \qquad \text{Eq.4.4}$$

With degree of freedom = (n-1)

Where

$$\sigma_s = \sqrt{\frac{\sum \left(x_i - \bar{x} \right)^2}{n-1}} \qquad \text{Eq.4.5}$$

Samples are collected from the people of age group 18-25 years and using null hypothesis, on the basis of statistical data and its analysis threshold level is calculated.

4.8 Chapter Summary

The chapter describes the reading and placement of the flex sensors used for system development as per the pre-defined standards. RFID tag extraction and LabVIEW GUI are explained with the help of flow charts and basic function used in the programming. Also the concept of hypothesis is discussed which for the analysis of statistical data collected from the samples and help to calculate the threshold value of flex sensors, which is explained in result analysis.