Research Inventy: International Journal of Engineering And Science
Vol.7, Issue 1 (Janaury 2017), PP -01-11
Issn (e): 2278-4721, Issn (p):2319-6483, www.researchinventy.com

Development and Application of a Failure Monitoring System by Using the Vibration and Location Information of Balises in Railway Signaling

Veysel Gökdemir
Selcuk University, Department of Electrical Electronics Engineering, Campus, Selcuklu, Konya, Turkey

Abstract: *In this project, a failure monitoring system is developed by using the vibration and location information of balises in railway signaling. A lot of field equipment in railway are loosening and broken in time period so that they need maintenance due to the vibrations that occur due to high speed trains traffic and railway vehicles impact. Among the field equipment, balises have very important role of communication in terms of transmitting information to trains. In this scope, it is aimed to make maintenance works more efficient, have no delayed trains, detect previously failure location and intervene in failure timely, by detecting and controlling balise cases such as loosening, out of place and the data consistency error that happens because of balise physical state. In this project, the communication is provided with I2C, Modbus RTU (Remote Terminal Unit) and RS485 standards by using Arduino Uno cards and MPU6050 IMU (Inertial Measurement Unit) sensors in laboratory. Each used sensors are in slave mode and computer interface designed with C# is in master mode. Fault situations in the system are checked instant by the interface. (it is assumed to mount the IMU sensor and the Arduino circuit on the balise) it is seen that the interface responds to the sensor movements instant and the system works well in the end of test processes.*

Keywords: *Accelerometer, balise, C# interface, gyroscope, Modbus RTU, RS485, railway signaling.*

I. Introduction

In the past, railway transportation systems have significantly provided transportation and hauling needs for people. Nowadays, railway systems are easing the burden of transportation because of increasing population of many countries. The railway systems that have high carrying capacity like high speed railway such as high speed trains, fast railway such as metros, suburban trains and normal railway such as light rail trams, freight trains offer transportation services as both environmental and economical thanks to especially their using of the electrical energy. The rail transportation and hauling are becoming important each day as a result of the rapid increase in the world's population. At the first years of the rail transportation, train movements had been provided by railway guards at previously set points. The accidents from human errors and mechanical faults had occurred because of the increasing of trains traffic. Thanks to technological developments, train line signals and switch points have been checked and train locations have been followed instant from far distance centers. Thus railway signaling systems have been created by directing train traffics from far distances. For the signaling systems, the rapid diagnosis system faults and the preventive maintenance are very important in terms of providing fast, comfort and secure voyages. In this scope, the features of balise fault monitoring system are briefly described below.

II. The System Design

In laboratory, an Arduino Uno and a MPU6050 IMU sensor are considered for each balise on train line, and each balise is in slave mode. The data received from these multi slaves are commented by the master mode pc interface designed with visual C#. The data received from the sensors arrives to the Arduino with I2C communication. This received data is forwarded to the remote master interface with the embedded codes in the Arduino, RS485 and Modbus RTU communications. The system general structure is shown in Fig. 1. (In reality the fiber optic communication is considered to use in the system).

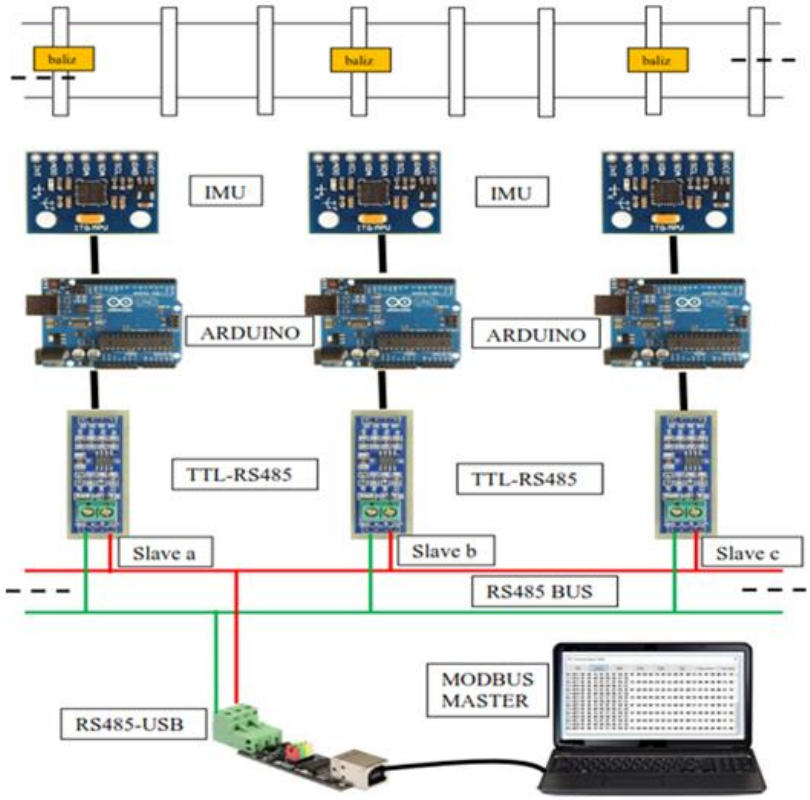

Fig. 1. The fault monitoring system general structure

2.1 Arduino Uno

Arduino Uno is a microcontroller card which has ATmega328P (Fig. 2). The Arduino has 14 digital input-output (6 of them can be used for PWM), 6 analog input, a 16 MHz quartz crystal, a USB connection, a ICSP connection and a reset button. Also it has 5V, 3.3V, GND and VIN pins in the power side. The Arduino's operation voltage is 5V. The input voltage can be between 7V and 12V. In this Project, USB connection is used for power voltage. The Arduino can run with a 9V adaptor.

Fig. 2. Arduino Uno R3

As memory features, the Arduino has Flash Memory 32KB, SRAM 2KB, EEPROM 1KB. Arduino IDE complier is used to compile the Arduino codes [1].

2.2 MPU6050 IMU Sensor

MPU6050 is an IMU sensor which has 3 axles accelerometer and 3 axles gyroscope (Fig. 3). Also it has a DMP digital motion processor and a heat sensor. The communication between IMU and Arduino is provided with I2C at 400 Hz. SCL and SDA connections provide I2C communication's logic level of VLOGIC reference pin. VLOGIC pin is connected to 3.3V pin because 5V pin can cause damage to the sensor. MPU-6050 converts the input-output of gyroscope and accelerometer to digital with 16bit ADC.

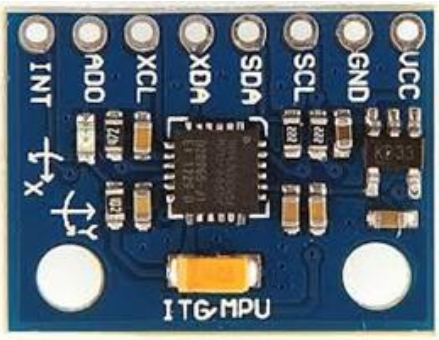

Fig. 3. MPU6050 IMU Sensor

2.2.1 Reading raw data

MPU-6050's 16-bit ADC feature simplifies reading the data by converting analog data to digital data. The output of the sensor consists of 16-bit, which is 8-bit low and 8-bit high, for each axle [2]. The sensor data, which is raw data and does not state any angle and acceleration, is received by the Arduino with I2C communication. The data is required to be converted to g force or angular velocity (°/s). At first ADC resolution is gained with (1) [3].

Resolution = Vref / 2^bit number (1)

The resolution is gained if the sensor ADC bit number is taken as 16 and reference voltage we used is taken as 3.3V. Required angular velocity or acceleration values are gained with (2).

(g or °/s) = (Raw data * Resolution - VzeroG) / Sensitivity (2)

The raw data is binary values received from accelerometer and gyroscope. There are VzeroG and sensitivity values for each sensor, which can be found in the catalogue of the sensor [3]. Table 1 shows the sensor's VzeroG and sensitivity values which are used in this project [2].

Table 1. MPU-6050 VzeroG and sensitivity values

Parameter	Accelerometer	Gyroscope
VzeroG	X and Y axles; ±50 mg Z axle; ±80 mg	±20 ° / s
Sensitivity	For ±2 g ; 16,384 LSB/g	For ±250°/s ; 131 LSB/(°/ s)

2.3 I2C

I2C (Inter-Integrated Circuit) has been developed by Philips in 1980s, which is a multi-controller serial data bus used to connect low speed peripheral units to mother board, embedded system or mobile phone. I2C communication protocol decreases the number of both pin and wire on PCB card by providing two cable connection, which are serial data (SDA) and serial clock (SCL), between devices and microcontrollers [4].

In this project, the communication between MPU6050 and the Arduino is provided with I2C protocol. Table 2 shows the connection pins between the sensor and the Arduino [1].

Table 2. Arduino Uno and MPU-6050 connection pins

Arduino Uno	MPU-6050
3.3V	VCC
GND	GND
Analog A4	SCL
Analog A5	SDA
Digital 2	INT

2.4 RS485

RS485 data bus forms a physical connection between master and slave devices. In this project, two cable half-duplex connection is used, which are A and B line. TTL-RS485 module with its max485 component provides a signal balance between the Arduino and RS485 bus. Logic 0 is obtained if B line is more positive than A line and logic 1 is obtained if A line is more positive than B. The voltage difference between A and B line should be minimum ±1.5V for an efficiency transmission. Input values in range ±200mV are in an undefined area. The RS485 bus, which has 100 kbps data speed for 1200 m, allows connection up to 1200 m and up to 32 slave devices [5]. In this project, baud rate value is used as 9600 bps for four sensors. Baud rate value is increased whenever distance and number of slave devices rise but in general it does not exceed 115200 bps, because this speed values become over for most microcontrollers and then systems begin to give faults [6].

Ra, Rc bias resistors and Rd termination resistor are very important to decrease the signal loses and the noise in RS485 bus connection (Fig. 4).

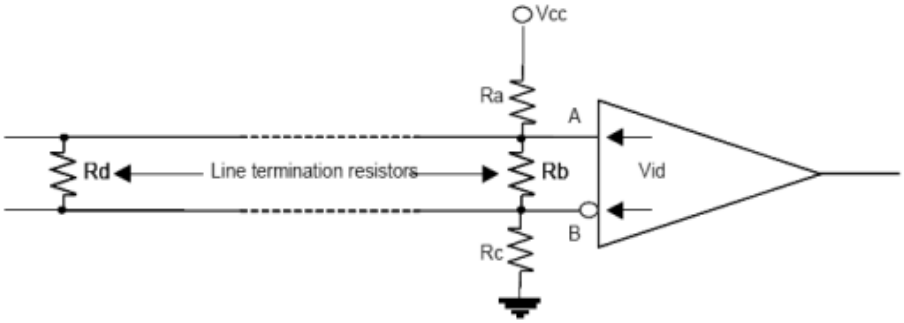

Figure 4. RS485 bias equivalent circuit

The cable characteristic impedance (Zo) which belongs to A and B line of RS485 bus is usually 100-120 Ω. (3) can be used to calculate bias values. Zo=Rb=Rd, Rt =Rb II Rd (Rb and Rd parallel), the voltage between A and B line (Vid) is taken as at least 200mV. Vcc can be taken as 5V. Ra and Rc are generally considered equal [7-8].

$$Vid=Vcc*(Rt / (Rt+Ra+Rc)) \quad (3)$$

In this project, Rd=Rb=120Ω, Ra=Rc=470Ω and Vcc=5V are taken for the system. When TTL-RS485 is used more than two devices, R7=120 Ω resistor on it is removed because of the bias values should be placed at the beginning and end of the line for RS485 bus [9].

2.5 Modbus

Modbus, which has been improved by AEG Modicon for PLCs in 1979, is an open source serial communication protocol. It is used as an industrial communication standard. Modbus applications can be implemented as serial line, TCP/IP and more fast networks (Modbus Plus). In applications on serial line, Modbus supports RS422 and RS485 as serial communication, master-slave as MAC protocol, RTU and ASCII as transmission mode. Master-slave protocol's processes are shown below [5].

- Communication is always started by master.
- A slave just responds on master's request.
- Slaves does not certainly communicate among themselves.
- Master just implements one process at same time.
- Master sends request to slaves as Unicast and Broadcast modes.

In Unicast mode, master addresses slave that it wants and just reply addressed slave. In Broadcast mode, master addresses all slaves on bus, all slaves begin writing processes by accepting requests, but no any response is sent to master. Modbus has 8 bit addressing size and 256 addresses. Address 0 is Broadcast mode address, 1-247 addresses are used as individual slave addresses [5].

In this project, Modbus RTU which provides more simplicity, more efficiency and takes less space is used. General transmission rate values used in Modbus protocol are 9600 bit/s or 19200 bit/s. In this project, the transmission rate value is chosen as 9600 bit/s.

2.5.1 Modbus coil-register structure

Information is stored in four different tables in slave devices (Table 3). Two of them hold on/off discrete values (coils), others hold numerical values. Each table can get 9999 values. Each coil or contact is 1 bit and their addresses are in the range 0000-270E. Each register is 1 word = 16 bit = 2 bytes, and their addresses are again in the range 0000-270E [10]. In this project, 30001-30006 registers are used because of only reading process in the system.

Table 3. Modbus coil/register table

Coil/Register	Data Addresses	Type	Table name
1-9999	0000-270E	Read Write	Discrete output coils
10001-19999	0000-270E	Only read	Discrete input contacts
30001-39999	0000-270E	Only read	Analog input registers
40001-49999	0000-270E	Only write	Analog output registers

2.5.2 Modbus message transmission

Modbus communication protocol uses a message framework for data traffic. This data framework is commented by master or slave devices, and a response is generated according to request. Modbus data framework consists of four parts (Table 4) [11].

Table 4. Modbus message framework

Slave Address	Function Code	Data	Error Control(CRC)
1 byte	1 byte	0-252 byte	2 byte CRC Low, CRC Hi

Modbus RTU message size is 256 bytes. When messages are transmitted, a space is put as at least 3.5 character in the beginning and end of the data framework among messages. So that slave is enabled to comment transmitted messages and to control errors. Slave addresses can get values in the range 1-247. Modbus has different function codes and these codes are determined according to which process will be done [12]. In this project, regBank.SetId( ), regBank.add( ) and regBank.set( ) functions in Modbus library are used.

2.6 Complementary Filter

As an alternative of kalman filter which is hard to understand and not practical, the complementary filter is used, which is simple to understand and implement. The task of this filter is to determine a more stable angle from multiple noisy signals not to impress signals. In this filter, signals are cleared from noises by passing through low and high filters. The sum of high and low filters is 1 and it is shown in (4).

$$G_1(s)+G_2(s)=1 \tag{4}$$

In this project, accelerometer and gyroscope values are passed through complementary filter (Fig. 5.) [13].

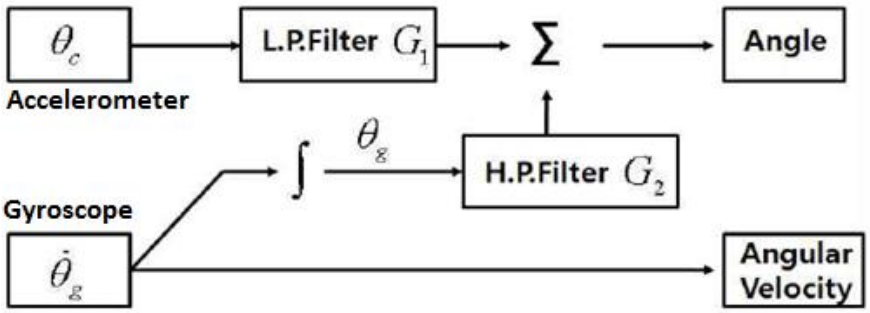

Fig. 5. Complementary filter

2.6.1 Complementary filter implementation to the Arduino

Implementation of complementary filter as software is quite simple. The filter composes of one digital high and low filter. Algorithm of the filter is shown in (5).

angleA=alpha*(gyroA*dt+get_last_A_angle())+ (1.0 - alpha) * accelA (5)

angleA: Obtained last angle,
alpha: Coefficient of high pass filter,
gyroA: Obtained angle from gyroscope,
accelA: Obtained angle from accelerometer,
get_last_A_angle (): Function to get last angleA,
dt: iteration loop time.
The alpha which is coefficient of the high pass filter is selected as 0.96 [14-15].

III. The System Interface Design

The system interface design is mentioned in this section. Subjects are processed respectively, which are working logic of the interface, graphical display of data in the interface and diagnosis fault in the interface.

3.1 The Interface Working Logic

Remote master interface determines whether there are faults in the system or not by getting the values of the sensors and follows data. The interface is designed with visual C# (Fig. 6) and its working logic is below.

- Collect the data.
- Compare this data with the reference values of balise in normal state
- Determine balise state.

Normal state: Balise is standing flat, not loose and in place.
Fault state:
Fault state 1 (Arıza durumu 1): Balise is not in place, there is too much slip.
Fault state 2 (Arıza durumu 2): Balise is too loose, vibration is too much.
Fault state 3 (Arıza durumu 3): Balise is not standing flat but curved.

- If there is a fault state in balises, show balises location, sticker and information of fault state.

For example; "Balise 01810 - km 200 - line 1 - Fault state 1"

- If there is normal state in balises, make bottom state display green, show fault number as 0. If there is a fault state in balises, make bottom state display red, show fault number.

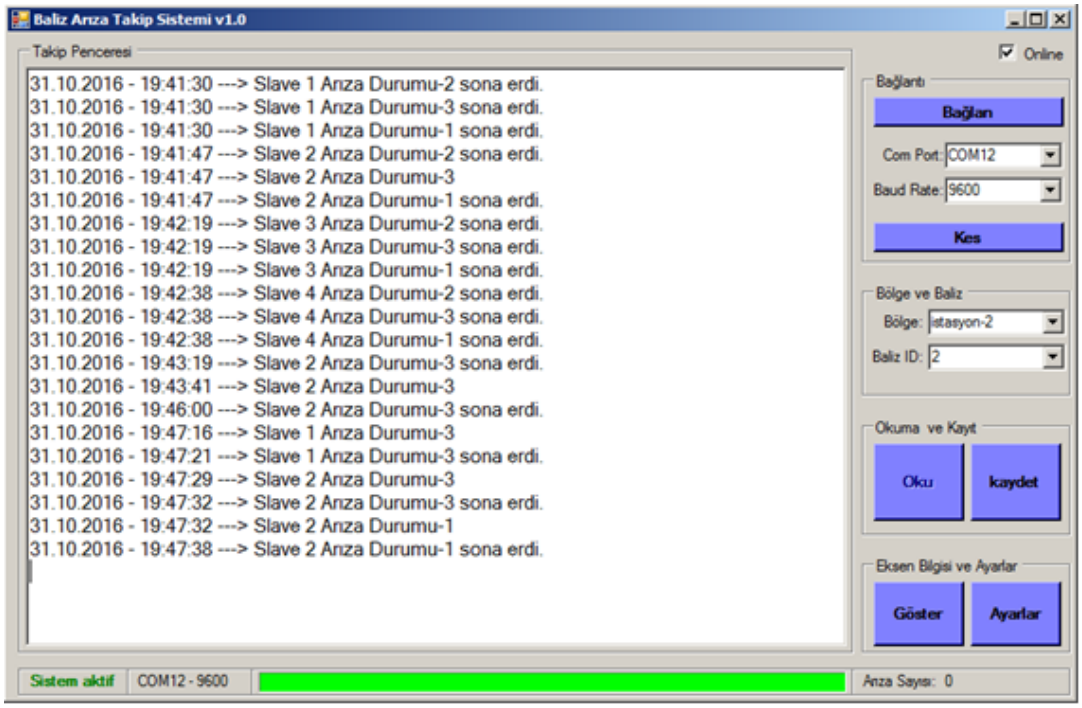

Fig. 6. The interface of the system that monitors the faults

3.2 The Data Graphical Display

The data belongs to each sensor can be checked as both visual and numerical with graphical display of the data. This case provides simplicity in terms of monitoring and controlling the data. For graphical display, dll that belongs to ZedGraph open source software is used. The window of data monitoring (Veri izleme) is opened when "Göster" button is pressed, which is in axle information and settings section (Eksen Bilgisi ve Ayarlar) in the interface (Fig. 7).

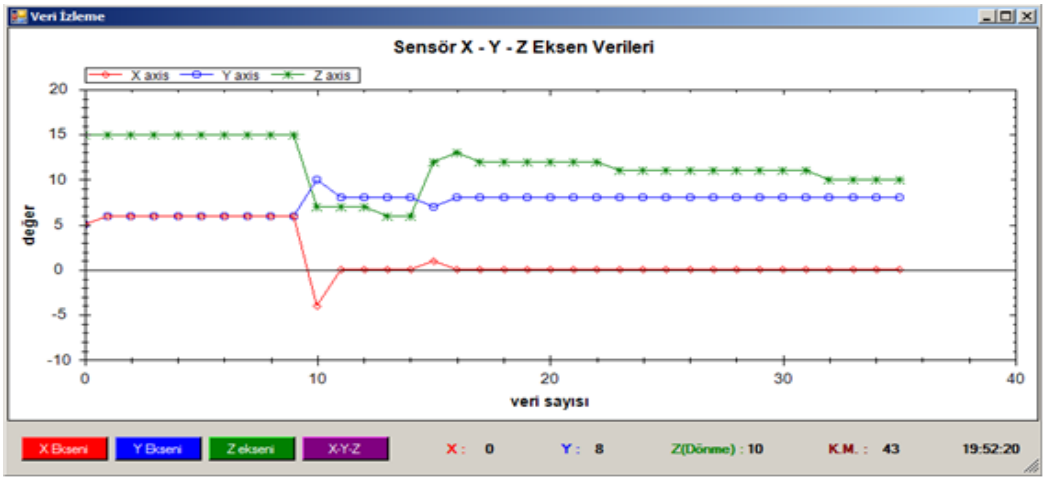

Fig. 7. Data monitoring window

3.3 Fault Diagnosis in The Interface

The gained data with the interface is commented according with MPU6050 sensor's x and y angles, z axle rotation values, x y z accelerometer values, and they are used for determining faults. Faults are divided into three states.

3.3.1 Fault state 1

This fault state means balise is not in place, there is too much sliding for balise place. For diagnosis of the fault state 1, the data received from accelerometer of the sensor is processed and predictive distance is calculated. This calculation logic is below.

- Read the data received from accelerometer in a specific "t" time.
- Accept first acceleration "a" value as constant.
- Get velocity with multiplication of acceleration changing value and time value ($v=\Delta a*t$).
- Multiply this velocity value and "t" time interval value, then find predictive sliding distance ($x=v*t$).

The algorithm used for this fault state in the interface looks like following.

```
if (KM  > Ref.val)
{
   Fault state 1 is available
}
else
{
   Fault state 1 ended
}
```

KM value is predictive sliding distance calculated with accelerometer values. Ref.val value is a reference value compared with KM. The calculation of KM value is below.

```
KM = register_gz - KRef;
```

register_gz value states sliding distance value that is calculated with accelerometer values in the Arduino software codes. KRef value is used to make KM value zero when there is a fault related sliding distance in the system and this fault is repaired. KRef value is recorded with "Kaydet" button on settings (Ayarlar) window (Fig. 8).

3.3.2 Fault state 2

This fault state means balise is too loose, the vibration of balise is too much. For diagnosis of the fault state 2, x and y angles values are used, which are obtained with complementary filter. The values of x and y angles are compared with specific reference values. The algorithm used for this fault state in the interface looks like following.

```
if ( register_ax > Ref.val || register_ax < -Ref.val || register_ay > Ref.val || register_ay < -Ref.val )
{
   Fault state 2 is available
}
else
{
   Fault state 2 ended
}
```

register_ax: is x angle value received from the sensor.
register_ay: is y angle value received from the sensor.
Ref.val: is a reference value that is used for comparing.

3.3.3 Fault state 3

This fault state means balise is not standing flat but curved. For diagnosis of the fault state 3, the data belongs to gyroscope z angle is used. This data is compared with specific reference values. The algorithm used for this fault state in the interface looks like following.

```
if ( R > Ref.val  ||  R < - Ref.val )
{
   Fault state 3 is available
}
else
{
   Fault state 3 ended
}
```

Ref.val value is a reference value compared with R value. The calculation of R value is below.

```
R=R_register-DRef;
```

R_register value is z angle rotation value received from gyroscope. DRef value is used to make R value zero when there is a fault related rotation in the system and this fault is repaired. DRef value is recorded with "Kaydet" button on settings (Ayarlar) window (Fig. 8).

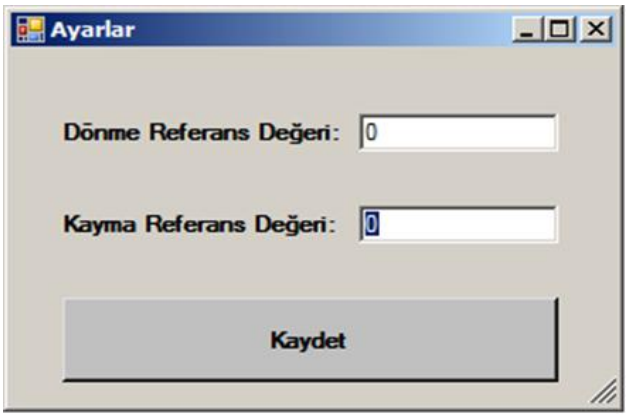

Fig. 8. Settings window

IV. The Interface Test Processes

In this section, the interface tests are processed. Before test processes, each MPU-6050 sensor is placed in a constant position to be and calibration settings are made without moving sensors (Fig. 9). Fig. 10 shows the system circuit board.

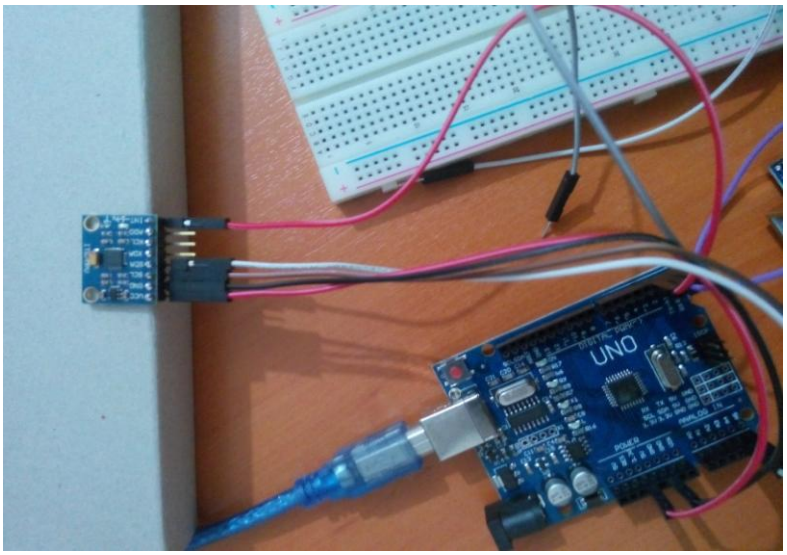

Fig. 9. MPU-6050 calibration settings

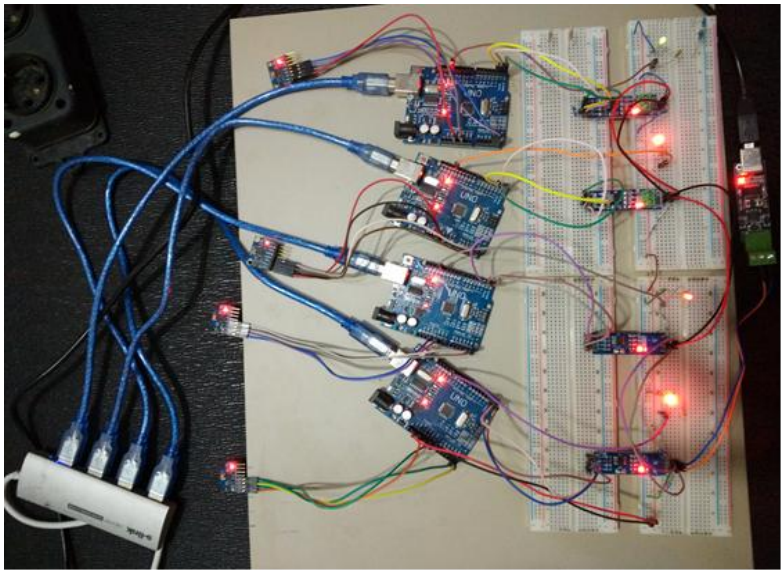

Fig. 10. The system circuit board

The Arduino calibration software version 1.1 is used for calibration settings [16]. Offset values are determined for the sensor when this software is loaded and run. The sensor offset values are respectively requested to close to 0 0 16384 0 0 0 for accelX accelY accelZ gyroX gyroY gyroZ. Raw value 16384 is gravity force which impacts on z axle of accelerometer when the sensor does not move. Then the determined values are used for each axle in set functions like below.

accelgyro.setXAccelOffset(-2736); accelgyro.setYAccelOffset(524);
accelgyro.setZAccelOffset(1026); accelgyro.setXGyroOffset(30);
accelgyro.setYGyroOffset(-51); accelgyro.setZGyroOffset(24);

For test processes, four MPU-6050 sensors, four Arduino Uno cards, four TTL-RS485 cards and a designed Modbus master interface are used. The system is connected to each sensor when it is run and while Online checkbox is selected. The data received from the sensors is checked and whether there is a fault or not in each balise is showed in the interface. Fig. 11 shows Online checkbox is selected. Each sensor state is scanned. There is no fault in each balise. Fault number is zero and fault state display is green. Fig. 12 shows that there is a fault in the system, fault number 1 and fault state display is red. In Fig. 13 the sensor is firstly moved from 0° to 90° then 90° to 0° at x axle. Then it is moved from 0° to -90° then -90° to 0° at y axle [17]. In Fig. 14 the sensor is firstly moved from 0° to -90° then -90° to 0° at y axle. Then it is moved from 0°to 90° then 90° to 0° at x axle [17].

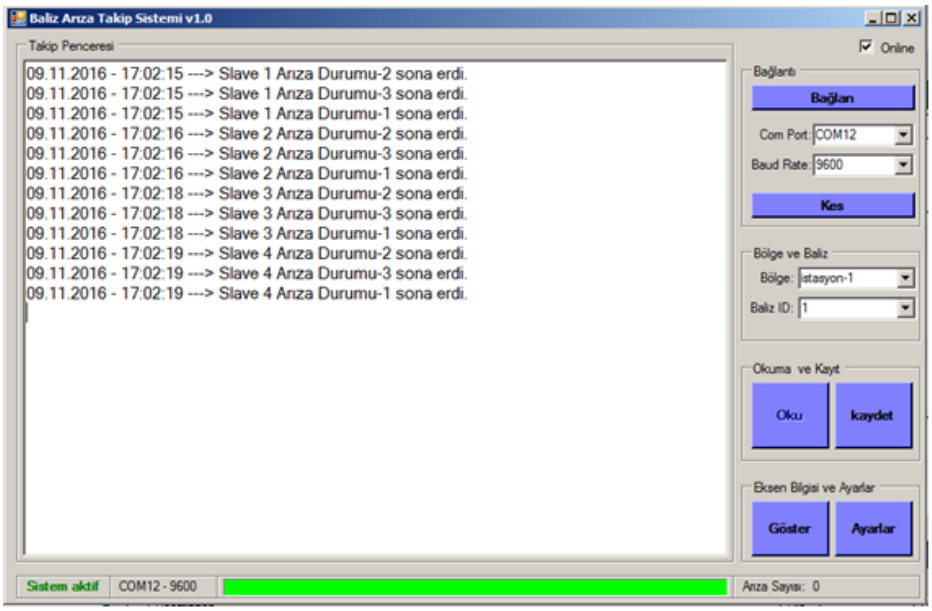

Fig. 11. There is no fault case in the interface.

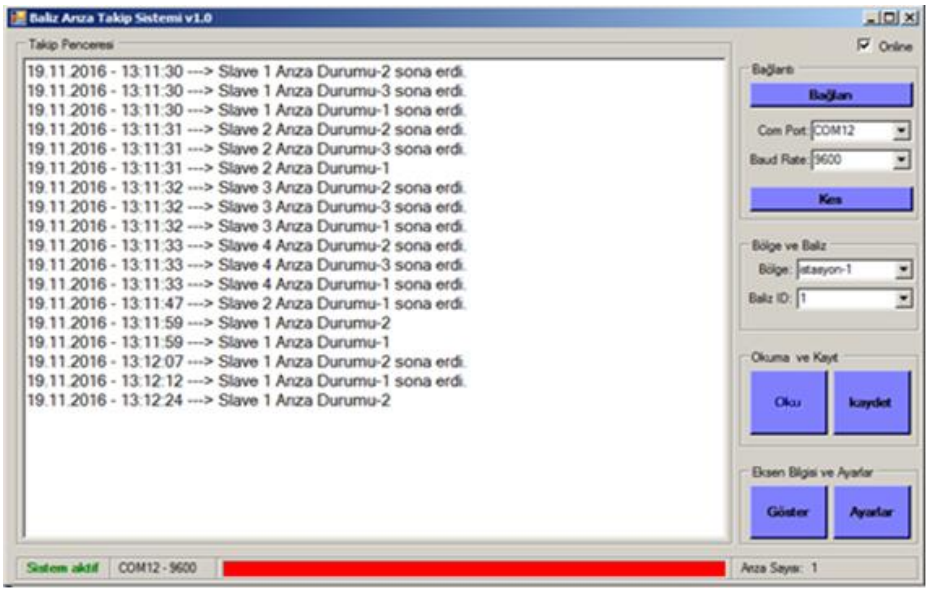

Fig. 12. There is a fault case in the interface

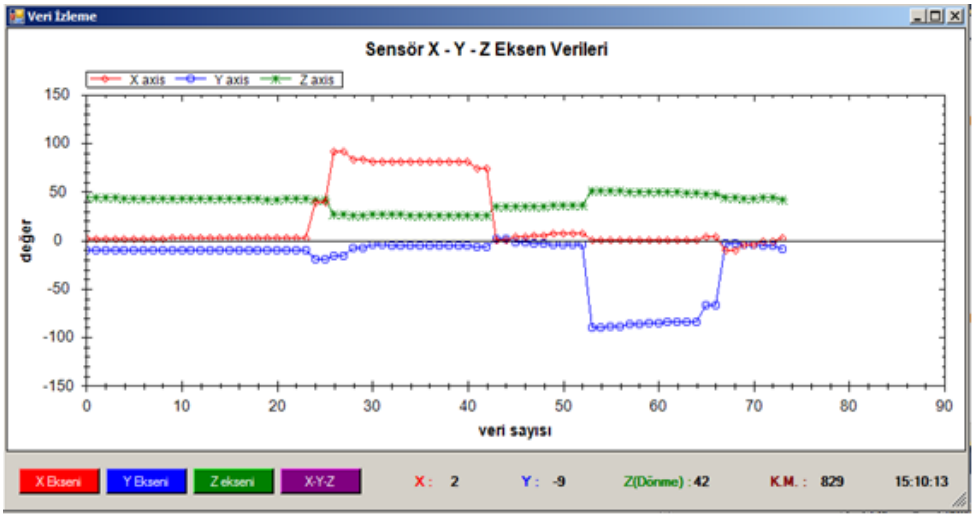

Fig. 13. The sensor 90° movements which are first at x axle then at y axle

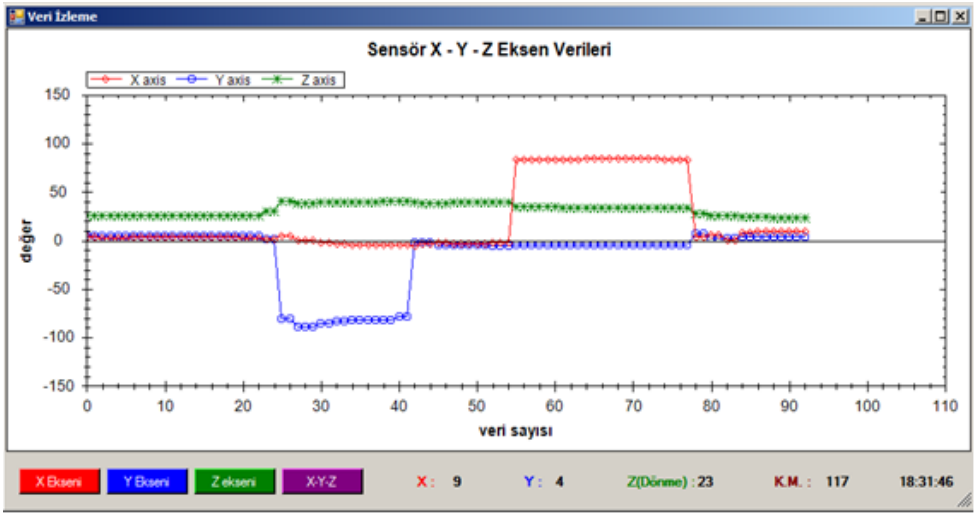

Fig. 14. The sensor 90° movements which are first at y axle then at x axle

V. Conclusion

In this project, the failure monitoring system is developed by using the vibration and location information of the balises in the railway signaling. The data from each slave is obtained with the designed Modbus master interface and RS485 Bus. The sensor test circuit is supposed to be integrated to the balise in test processes and then the responses of the sensors are tried to be measured as the train passes on the railway. When the sensor is moved, the interface replies instant and the data of the sensor is achieved and monitored graphically in the interface.

This project is advantageous in terms of the instant controlling of balises which are placed in railway signaling systems, previously diagnosis physical balise faults (such as out of place, loosening, be broken), fault repair in short time and the direct arrival of maintenance teams to fault location according to the clear fault report by CTC (Central Traffic Controller), the reducing delaying times that the balises cause, which are the signaling equipment of level 1, level 2 and level 3 in ERTMS/ETCS, the traveling of passengers without the delaying time.

On the other side, as we look at railway lines lengths and used balises numbers, the system is seemed to be disadvantageous in term of the system integration to each balise, the length of the used Modbus RTU and the RS485 communication which can be used up to distance of 1200 m, the need to use amplificator circuits after 1200 m and maybe the cost of the system.

When above the advantages and the disadvantages are considered, railway institutes open high cost tenders to private firms for maintenance works because of sustaining trains traffic without faults and decreasing delaying times, the system can be integrated to the current fiber optic lines which are placed in railway lines for signaling systems as the hardware, software and communication of this system are revised and arranged, thus the importance of this and similar projects will be comprehended better.

References

[1]. Arduino, Arduino UNO & Genuino UNO, https://www.arduino.cc/en/Main/ArduinoBoardUno, 2016.

[2]. InvenSense Inc., MPU-6000 and MPU-6050 Product specification revision 3.4, document number: PS- MPU-6000A-00, U.S.A, 2013.

[3]. Starlino, *A Guide to using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications*, http://www.starlino.com/imu_guide.html, 2009.

[4]. S.K. Jose, X.A. Mary and N. Mathew, ARM 7 Based Accident Alert and Vehicle Tracking System, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-2, Issue-4, March 2013.

[5]. Dr. H.K. Verma, *Sensor Networks*, Sharda University, Greater Noida, 2014.

[6]. J. Lindblom, *Serial communication*, https://learn.sparkfun.com/tutorials/serial-communication#rules-of-serial, 2012.

[7]. STMicroelectronics, AN1690 Application note, 2007.

[8]. L. Zhao, R. Liang and J. Zhang, 2015, Solving for the best value of bias resistor to promote stability of RS485 fieldbus, International Journal of Future Generation Communication and Networking Vol. 8, No. 3 (2015), pp. 89-96.

[9]. T. King, RS485 serial communications; RS485 module, http://arduino-info.wikispaces.com/RS485-Modules, 2015.

[10]. Simply Modbus, How is data stored in Standard Modbus? And Exception Responses, http:// www.simplymodbus.ca/FAQ.htm and http:// www.simplymodbus.ca/exceptions.htm, 2015.

[11]. Z. Hao, L. Guohuan, W. Honghui and S. Zhongkui, 2012, Development for protocol conversion gateway of Industrial Field Bus, International Conference on WTCS 2009, AISC 117, pp. 211–216.

[12]. Modbus.org, MODBUS over Serial Line Specification & Implementation guide V1.0., 2002.

[13]. H.G. Min, E.T. Jeung, Complementary Filter Design for Angle Estimation using MEMS Accelerometer and Gyroscope, Robotics Lab., NTREX Ltd., co., Juan-Dong 5-38, Nam-Gu, Incheon, Korea, Department of Control and Instrumentation, Changwon National University, Changwon, 641-773, Korea.

[14]. Starlino, A Guide to using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications, http://www.starlino.com/imu_guide.html, 2009.

[15]. J. Rowberg, Arduino sketch for MPU6050 to calculate commplementary filter data, 2012.

[16]. L. Rodenas, Arduino sketch that returns calibration offsets for MPU6050 Version 1.1., 2014.

[17]. S.A. Juliusdottir, Movement measurement device for airplanes, Haskolinn Reykjavik University, 2014.