International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:8, No:7, 2014

Design of a Low Cost Motion Data Acquisition Setup for Mechatronic Systems

Barış Can Yalçın

***Abstract*—**Motion sensors have been commonly used as a valuable component in mechatronic systems, however, many mechatronic designs and applications that need motion sensors cost enormous amount of money, especially high-tech systems. Design of a software for communication protocol between data acquisition card and motion sensor is another issue that has to be solved. This study presents how to design a low cost motion data acquisition setup consisting of MPU 6050 motion sensor (gyro and accelerometer in 3 axes) and Arduino Mega2560 microcontroller. Design parameters are calibration of the sensor, identification and communication between sensor and data acquisition card, interpretation of data collected by the sensor.

***Keywords*—**Calibration of sensors, data acquisition.

I. Introduction

OVER the last few decades, motion sensors have become more commonly popular in industrial usage. The application areas of these sensors are automotive, space, robotic, manufacturing, biomedical, smart phones [1]-[3]. Specific application areas are given below [4];

- BlurFree™ technology (for Video/Still Image Stabilization)
- AirSign™ technology (for Security/Authentication)
- TouchAnywhere™ technology (for "no touch" UI Application Control/Navigation)
- MotionCommand™ technology (for Gesture Short-cuts)
- Motion-enabled game and application framework
- InstantGesture™ iG™ gesture recognition
- Location based services, points of interest, and dead reckoning
- Handset and portable gaming
- Motion-based game controllers
- 3D remote controls for Internet connected DTVs and set top boxes, 3D mice
- Wearable sensors for health, fitness and sports
- Toys

In control processes of rigid body motion, the information about translational and angular position is very important, because of these parameters are used to calculate control signal into plant. For calculating the input signal accurately, calibration of the motion sensor and interpretation of data that came from the sensor has to be calculated well. Actually there are many contributions onto solve these problems in literature; however, most of them need an expensive mechatronic setup.

B. C. Yalçın is with Yildiz Technical University, Istanbul, Turkey (e-mail: bcyalcin@yildiz.edu.tr).

In this study, this problem is solved using a low cost motion sensor, which is MPU 6050, and a low cost data acquisition card, which is Arduino Mega2560. Main specifications of MPU 6050 are;

- Selectable +-2/4/8/16g accelerometer range
- Selectable +-250/500/1000/2000 degrees/s gyroscope range
- 16 bit output from both sensors
- Gyroscope linear acceleration sensitivity of 0.1 degrees/s, a vast improvement over the three axis gyroscopes of other companies.
- Low noise on both outputs, see the datasheet
- Data output rate up to 1000Hz
- Digital low pass filter has a maximum corner frequency of 256Hz.

The main communication loop between process and equipment (sensor and daq card) is described in Fig. 1 below.

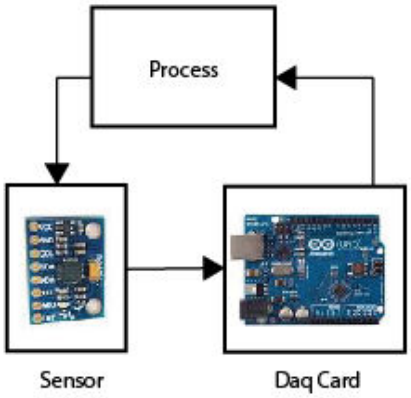

Fig. 1 Communication loop

For power supply flexibility, the MPU-6050 operates from VDD power supply voltage range of 2.375V-3.46V. Additionally, the MPU-6050 provides a VLOGIC reference pin (in addition to its analog supply pin: VDD), which sets the logic levels of its I2C interface. The VLOGIC voltage may be 1.8V±5% or VDD. Data acquisition systems, as the name implies, are products and/or processes used to collect information to register or analyze some process. In the simplest form, a technician logging the temperature of an oven on a piece of paper is performing data acquisition. As technology has progressed, this type of process has been simplified and made more accurate, versatile, and reliable through electronic equipment. Equipment ranges from simple recorders to sophisticated computer systems. Data acquisition

products serve as a focal point in a system, tying together a wide variety of products, such as sensors that indicate temperature, flow, level, or pressure.

Generally engineers are prone to use an expensive setup-up that is produced by well-known companies for their design problems, the main reason of this situation is that, those companies support their products after sale, thus engineers accept the cost of whole process. The basic contribution of this work is to show people most of engineer problems that require a motion sensor and a data acquisition card can be solved in a less expensive way.

In second section, how to program MPU6050 for calibration process is described. In third section, identification and communication protocol between MPU6050 and Arduino Mega2560 is conducted. In fourth section, interpretation of data collected by sensor is given.

II. Calibration Process of MPU6050

MPU6050 motion sensor contains 3 angular axis accelerometer and 3 angular axis gyroscopes. It converts 16 bit analog-digital converter for digitizing accelerometer and gyroscope values using its onboard Digital Motion Processor™. For tracking fast and slow motions of a rigid body, angular position detection scale can be programmed to the values of ±250°/sec, ±500°/sec, ±1000°/sec and ±2000°/sec, angular acceleration detection scale can be programmed to the values of ±2 *g*, ±4 *g*, ±8 *g*, ±16 *g*. Almost every sensor have some manufacturing errors, because of this reason, calibration process has to be done. There are two parameters that need to be adjusted in calibration process. First one is gain, second one is offset. To adjust gain and offset, sensor has to be in a stable position. *FreeIMU* [5] is one of the most popular libraries for this task. Using it with Arduino's official software, calibration task can be done easily.

The Arduino takes sensor readings for adjusted seconds during the process, and tracks the highest and lowest values it obtains. These sensor readings during the seconds of the process define the minimum and maximum of expected values for the readings taken during the loop. Meanwhile, Arduino knows that the sensor is stable and minimum and maximum values the sensor reaches actually are interference. This information is recorded by Arduino, and it is being used when the sensor works during the process. The calibration process steps are given in 9 steps below;

1. Locate the sensor on a stable structure
2. Start the sensor (give it energy)
3. Start the algorithm via daq card
4. The sensor starts collecting information for fixing interference
5. Stop the algorithm
6. Stop the sensor
7. Make the necessary connections between the process and equipment (sensor and daq card)
8. Start the sensor
9. Start the data acquisition card

And recorded interference data during calibration process can be seen from Fig. 2 below.

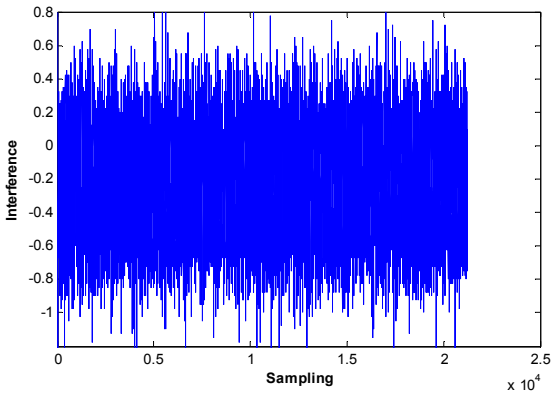

Fig. 2 Recorded interference

III. Identification and Communication between Sensor and Data Acquisition Card

Identification of sensor to Arduino Mega2560 is achieved using wire and MPU6050 libraries. These libraries exist in *Processing 2.0.3*'s library folder [5], flow diagram of identification process is given in Fig. 3 and to use these library's, the code shown in Fig. 4 can be used in Arduino's official software.

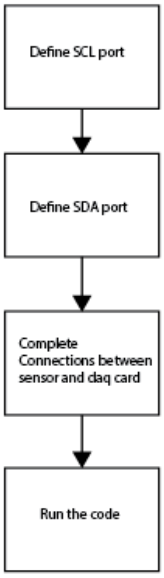

Fig. 3 Flow diagram of identification process

```
#include "Wire.h"
#include "MPU6050.h"

MPU6050 mpu;
int ax, ay, az, gx, gy, gz;
float smooth_ax, smooth_ay, smooth_az;

void setup()
{
  Wire.begin();
  Serial.begin(38400);
  mpu.wakeup();
}

void loop()
{
  mpu.read6dof(&ax, &ay, &az, &gx, &gy, &gz);

  smooth_ax = 0.95 * smooth_ax + 0.05 * ax;
  smooth_ay = 0.95 * smooth_ay + 0.05 * ay;
  smooth_az = 0.95 * smooth_az + 0.05 * az;

  Serial.print(smooth_ax);
  Serial.print("\t");
  Serial.print(smooth_ay);
  Serial.print("\t");
  Serial.print(smooth_az);
  Serial.print("\t");
  Serial.print(gx);
  Serial.print("\t");
  Serial.print(gy);
  Serial.print("\t");
  Serial.println(gz);
}
```

Fig. 4 Communication code

Communication between sensor and data acquisition card is achieved using *Serial Clock Line* (SCL) and *Serial Data Line* (SDA), those protocols are both the layers of *Inter Integrated Circuit*, I^2C uses these two bidirectional open-drain lines. Typical voltages are +5 V and +3.3 V. SCL and SDA ports on MPU6050 and Arduino Mega6050 can be seen in Figs. 5 and 6.

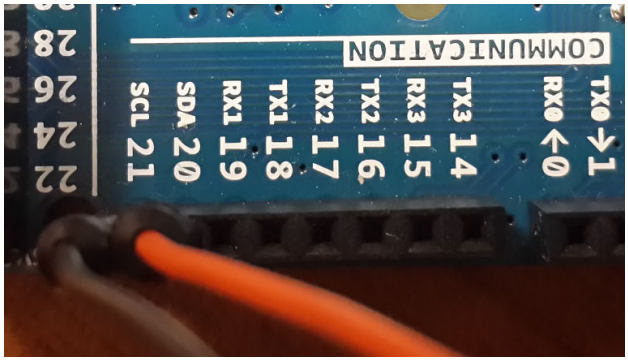

Fig. 5 SCL and SDA ports on Arduino Mega2560

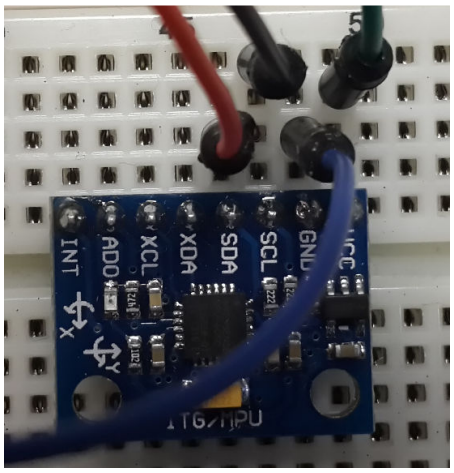

Fig. 6 SCL and SDA ports on MPU6050

The address and the data bytes are sent most significant bit first. The start bit is indicated by a high-to-low transition of SDA with SCL high; the stop bit is indicated by a low-to-high transition of SDA with SCL high. All other transitions of SDA take place with SCL low.

IV. Interpretation of Data Collected by the Sensor

The data that come from MPU6050 is raw and it has to be processed for an analog shape. To obtain raw data, the code given in Fig. 7 can be used. In Fig. 8, first three columns represent raw angular acceleration values while last three columns represent raw angular position values changing in time.

```
LDByteReadI2C(MPU6050_ADDRESS, MPU6050_RA_ACCEL_XOUT_H, &ACCEL_XOUT_H, 1);
      LDByteReadI2C(MPU6050_ADDRESS,
                    MPU6050_RA_ACCEL_XOUT_L, &ACCEL_XOUT_L, 1);
      LDByteReadI2C(MPU6050_ADDRESS,
                    MPU6050_RA_ACCEL_YOUT_H, &ACCEL_YOUT_H, 1);
      LDByteReadI2C(MPU6050_ADDRESS,
                    MPU6050_RA_ACCEL_YOUT_L, &ACCEL_YOUT_L, 1);
      LDByteReadI2C(MPU6050_ADDRESS,
                    MPU6050_RA_ACCEL_ZOUT_H, &ACCEL_ZOUT_H, 1);
      LDByteReadI2C(MPU6050_ADDRESS,
                    MPU6050_RA_ACCEL_ZOUT_L, &ACCEL_ZOUT_L, 1);

      ACCEL_XOUT = ((ACCEL_XOUT_H<<8)|ACCEL_XOUT_L);
      ACCEL_YOUT = ((ACCEL_YOUT_H<<8)|ACCEL_YOUT_L);
      ACCEL_ZOUT = ((ACCEL_ZOUT_H<<8)|ACCEL_ZOUT_L);
```

Fig. 7 The code for raw data

8140	14200	-1748	49	76	-226
8148	14184	-1824	48	97	-235
8188	14232	-1680	44	86	-209
8228	14196	-1796	64	89	-228
8180	14164	-1756	28	71	-201
8216	14060	-1896	17	66	-207
8264	14244	-1868	95	54	-280
8088	14284	-1896	46	70	-249
8228	14196	-1908	34	61	-222
8176	14228	-1856	49	58	-243
8196	14192	-1796	21	75	-198
8176	14156	-1900	37	81	-218
8216	14268	-1760	55	66	-261
8232	14320	-1880	67	73	-241
8216	14204	-1772	9	101	-192
8176	14176	-1732	8	96	-200
8232	14192	-1760	32	108	-186

Fig. 8 Raw data

To understand meanings of raw data, angular position and angular acceleration scale adjustment have to be known. The code to achieve this task is given in Fig. 9 below. For detailed information, [6] has to be read.

```
void Get_Gyro_Rates()
{
        LDByteReadI2C
(MPU6050_ADDRESS, MPU6050_RA_GYRO_XOUT_H, &GYRO_XOUT_H, 1);
        LDByteReadI2C
(MPU6050_ADDRESS, MPU6050_RA_GYRO_XOUT_L, &GYRO_XOUT_L, 1);
        LDByteReadI2C
(MPU6050_ADDRESS, MPU6050_RA_GYRO_YOUT_H, &GYRO_YOUT_H, 1);
        LDByteReadI2C
(MPU6050_ADDRESS, MPU6050_RA_GYRO_YOUT_L, &GYRO_YOUT_L, 1);
        LDByteReadI2C
(MPU6050_ADDRESS, MPU6050_RA_GYRO_ZOUT_H, &GYRO_ZOUT_H, 1);
        LDByteReadI2C
(MPU6050_ADDRESS, MPU6050_RA_GYRO_ZOUT_L, &GYRO_ZOUT_L, 1);

        GYRO_XOUT = ((GYRO_XOUT_H<<8)|GYRO_XOUT_L) - GYRO_XOUT_OFFSET;
        GYRO_YOUT = ((GYRO_YOUT_H<<8)|GYRO_YOUT_L) - GYRO_YOUT_OFFSET;
        GYRO_ZOUT = ((GYRO_ZOUT_H<<8)|GYRO_ZOUT_L) - GYRO_ZOUT_OFFSET;

        GYRO_XRATE = (float)GYRO_XOUT/gyro_xsensitivity;
        GYRO_YRATE = (float)GYRO_YOUT/gyro_ysensitivity;
        GYRO_ZRATE = (float)GYRO_ZOUT/gyro_zsensitivity;
```

Fig. 9 Code for meaningful data

After applying the code given in Fig. 9, meaningful data for sensor parameters (position and acceleration data) can be obtained, these data can be seen in Fig. 10 below, first three columns represent angular acceleration values while last three columns represent angular position values changing in time.

41.2	30.2	10.1	1.2	2.1	1.2
11.2	29.2	19.1	3.2	6.2	2.6
31.2	29.2	31.1	2.3	1.6	4.4
43.2	10.5	23.1	6.2	2.4	8.2
61.2	30.4	61.1	2.6	7.2	7.4
42.2	20.3	12.1	1.5	1.6	6.3
31.2	10.2	15.1	1.9	3.5	5.2
48.2	15.2	13.1	7.7	2.7	1.1
45.2	18.2	61.1	2.5	1.4	3.7
42.2	23.9	21.1	6.3	6.2	2.5
41.2	30.3	12.1	1.2	7.2	6.3
39.2	19.1	61.1	2.1	3.1	2.2

Fig. 10 Meaningful data

V. Conclusion

As a conclusion, a low cost data acquisition setup is created using MPU 6050 motion sensor and Arduino Mega2560. Communication between Arduino and MPU 6050 is achieved. According to the sensor readings for adjusted seconds during the process, its highest and lowest values are obtained; calibration process is done to remove interference.

How to program MPU6050 for calibration process is described, identification and communication protocol between MPU6050 and Arduino Mega2560 is conducted and interpretation of data collected by sensor is given.

References

[1] Yamada, K. and M. Soga, *A compact integrated visual motion sensor for ITS applications.* Intelligent Transportation Systems, IEEE Transactions on, 2003. **4**(1): p. 35-42.
[2] Sandborn, P.A.M. and P. Abshire. *2D motion sensor with programmable feature extraction*. in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*. 2013.
[3] Kamnik, R., S. Stegel, and M. Munih. *Design and Calibration of Three-Axial Inertial Motion Sensor*. in *Power Electronics and Motion Control Conference, 2006. EPE-PEMC 2006. 12th International*. 2006.
[4] http://invensense.com/mems/gyro/documents/RM-MPU-6000A.pdf.
[5] http://www.varesano.net/projects/hardware/FreeIMU.
[6] https://processing.org.