AXZON

Application Note AN011
Sensor Code Algorithm

1 Introduction

Sensor tags based on Magnus®-S3 produce a Sensor Code which gives information about the sensor tag's environment. A change in a tag's environment is usually detected by a change in its sensor code. To maximize the precision of sensor measurements, the user should understand and account for effects related to the reader transmission frequency, the amount of power received by the sensor tag, averaging, and command timing.

Magnus®-S3 Sensor Code value is frequency dependent. There are three recommended techniques to deal with this frequency dependency:

1. Use a single frequency. Regulatory requirements prevent continuous transmission at a single frequency; compliance will limit the sample rate. Therefore, the use of this technique in real applications is limited
2. Use the average Sensor Code value over the entire frequency band. The frequency range must be sampled adequately and evenly to avoid biasing the result. This technique can only be used when it is possible to collect enough data
3. Use regression analysis to fit the Sensor Code values to a line, then take the value of the line at some fixed frequency. This technique provides a faster and more accurate measurement but it is more complex to implement

This note describes one algorithm that is based on regression analysis technique and provides an example that can be used to calculate a change in the sensor code to reflect the change in the tag's environment when the data collected by the reader is noisy.

2 Pseudo Code Implementation

1. Take Sensor Code readings, while frequency hopping, until the reader collects **valid** readings:
    a. at N or more distinct frequencies with values between 32 and 464 or
    b. at N or more distinct frequencies with values less than or equal to 32 or
    c. at N or more distinct frequencies with values greater than or equal to 464
2. If the reader was unable to perform enough valid readings to fulfill any of the three previous conditions before a defined timeout, then report no measurement. Go to End (step 10)
3. If there are N or more readings greater than 32 and less than 464 then go to step 6
4. If there are N or more readings less than or equal to 32 then
    a. Sensor Code Measurement = average (all readings, including the ones greater than 32)
    b. Go to End (step 10)
5. If there are N or more readings greater than or equal to 464 then
    a. Sensor Code Measurement = average (all readings, including the ones less than 464)
    b. Go to End (step 10)
6. Discard all readings that are less than or equal to 32 and all readings that are greater than or equal to 464 and perform a linear regression analysis to find slope and intercept
7. Calculate the Sensor Code Measurement at the middle of the frequency range (i.e. 915 MHz for FCC) using the slope and intercept
8. If Sensor Code Measurement is greater than 491 then Sensor Code Measurement = 491
9. If Sensor Code Measurement is less than 5 then Sensor Code Measurement = 5
10. End

3 Notes on Pseudo Code Implementation

3.1 Notes About Step 1

- The higher the value of N the better; it will increase accuracy but it will require more time to collect the readings. The minimum recommended value for N is 5. But this may not always be possible, so the absolute minimum value for N is 3
- Some readers may return more readings than N. More readings will provide a better measurement, so these additional readings must be kept
- If there are two or more readings performed at the same frequency, then these values must be averaged and count as a single reading
- Valid Readings
  - The Sensor Code reading for Magnus-S3 is a nine-bit value. Nine bits provide a range from 0 to 511. But Magnus-S3 will only produce Sensor Code readings in the range from 5 to 491
  - Readings with a value that is less than or equal to zero or greater than 511 are not valid and must be discarded
  - Readings with a value of zero must be discarded for two reasons: because it is not valid (Magnus-S3 cannot produce it) and because some reader API's mishandle error conditions in the communication protocol and return a value of zero when the reader just could not read the memory location corresponding to the Sensor Code
  - The Sensor Code readings must be performed at the specified On-Chip RSSI range. The general recommendation for the On-Chip RSSI range is from 3 to 15. Discard any readings with an On-Chip RSSI outside this range

3.2 Note About Step 2

- There is no recommendation for the timeout value. This time depends on the reader performance, the value of N and the specifics of the application

3.3 Note About Step 4 and 5

- These two steps handle the case when the Sensor Code is saturated. In this case the average operation provides a better indication of the value of the Sensor Code instead of the regression analysis

3.4 Note About Step 6

- Measurements under 32 and over 464 indicate Sensor Code saturation. Therefore, these measurements are outside the linear zone of the Sensor Code and should not be used in the regression analysis

3.5 Note About Steps 8 and 9

- The regression analysis can produce Sensor Code values outside the normal range. But these values still indicate saturation. Therefore, the resulting Sensor Code should be limited to the range 5 to 491

3.6 Linear Regression Analysis Calculations for Steps 6 through 9

Following there is a C# implementation of a linear regression analysis:

```
double[] Frequencies; // Array of all the frequency values
double[] SensorCodes; // Array of all the Sensor Code values
int arrayLen = 0;     // Length of the previous two arrays

double slope = 0;      // Resulting slope of the regression analysis
double yintercept = 0; // Resulting yintercept of the regression analysis

////////////////////////////////////////////////////////////////////////////
//
// HERE GOES THE CODE THAT IMPLEMENTS STEPS 1 THROUGH 5
// THIS CODE UPDATES THE ARRAYS(Frequencies and SensorCodes) AND arrayLen
//
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
// HERE GOES THE CODE THAT DISCARDS READINGS THAT ARE LESS THAN OR EQUAL TO
// 32 AND GREATER THAN OR EQUAL TO 464
//
////////////////////////////////////////////////////////////////////////////

// Linear regression analysis
double SumX = 0;
double SumY = 0;
double SumXY = 0;
double SumXX = 0;
for (int i = 0; i < arrayLen; i++)
{
    SumX = SumX + Frequencies[i];
    SumY = SumY + SensorCodes[i];
    SumXY = SumXY + Frequencies[i] * SensorCodes[i];
    SumXX = SumXX + Frequencies[i] * Frequencies[i];
}
slope = (arrayLen * SumXY - SumX * SumY) / (arrayLen * SumXX - SumX * SumX);
yintercept = (SumY - slope * SumX) / arrayLen;

// Most reader APIs handle frequencies in kHz so adjust next value
// accordingly. Also adjust value for the frequency band
double MidFrequency = 915000;  // Example: 915 MHz for FCC band
double SensorCodeMeasurement = MidFrequency * slope + yintercept;

if (SensorCodeMeasurement > 491) SensorCodeMeasurement = 491;
if (SensorCodeMeasurement < 5) SensorCodeMeasurement = 5;
```

4 Example

The example below is for a tag that is placed inside a vehicle to measure water leakage into the car. The sensor code of the tag changes depending on the amount of water that is next to it. The data was first collected for a “Dry” tag and then followed by a “Wet Low” and “Wet Medium” indicating low amount of water and medium amount of water for the two wet measurements.

- Use a Quadratic Curve Fitting for the Baseline Measurement (Dry)
    - Get at least four measurements making sure that at least one measurement is under 915 MHz and at least one measurement over 915 Mhz. This is shown in Figure 1
    - Save the equation coefficients

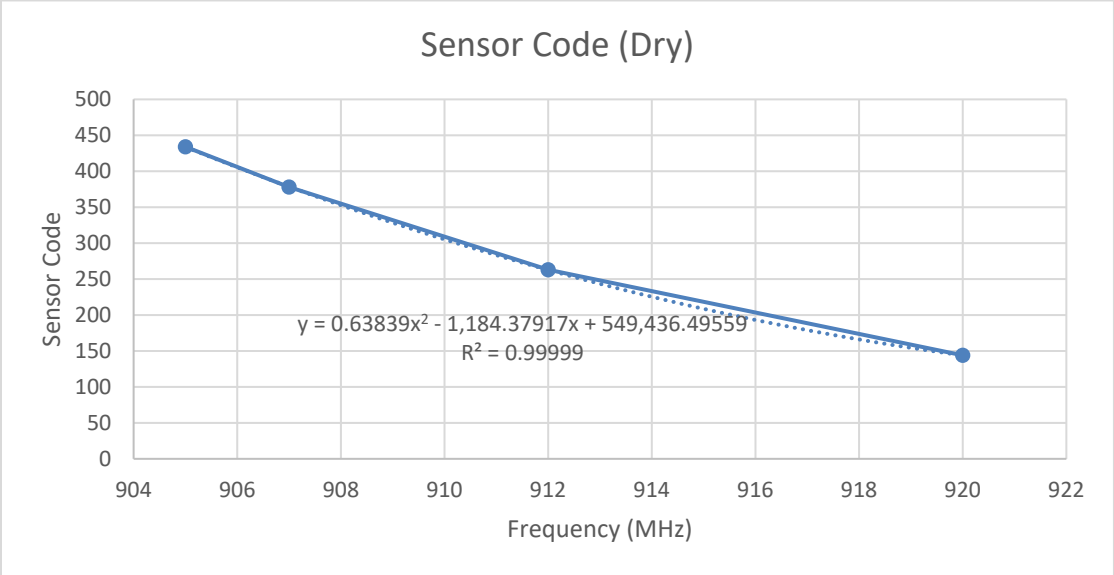

Figure 1 Curve fitting of the Sensor Codes for the "Dry" conditions.

- After wetting the tags, retrieve the coefficients
    - In theory only one Sensor Code measurement is needed (two or more can add confidence). Preferably the measurement should be done under 915 MHz where there is more Sensor Code difference
    - Use the frequency of the Sensor Code just measured in the quadratic equation
    - Compare the calculated “dry” Sensor Code with the one just measured using a predefined threshold
    - It should be possible to distinguish very low changes in Sensor Code, maybe as low as 5 codes.

The dry measurement as well as the “Wet Low” and “Wet Medium” are shown in Figure 2.

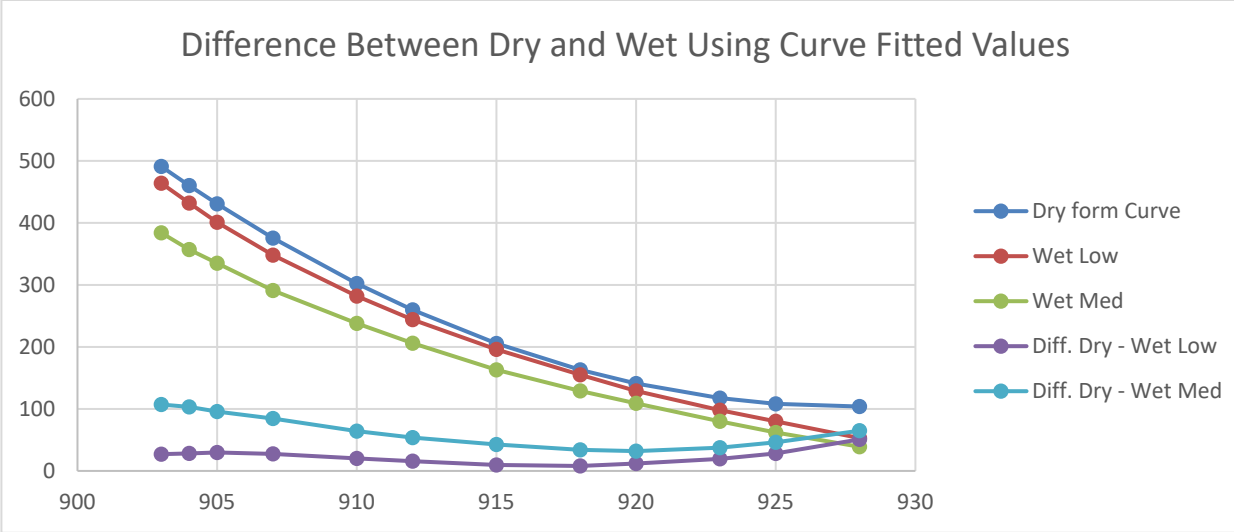

Figure 2 Difference between Dry, Wet Low and Wet Medium using Curve Fitted values

5 Conclusions

Curve Fitting techniques look promising in detecting small changes in the Sensor Code and with a small number of individual measurements, which is convenient for an application where measurement time is critical.

With this method, it should be possible to distinguish very low changes in Sensor Code, maybe as low as 5 codes.

6 Notices

RFMicron, Inc., ("RFMicron") conditionally delivers this document to a single, authorized customer ("Customer"). Neither receipt nor possession hereof confer or transfers any rights in, or grants any license to, the subject matter of any drawings, designs, or technical information contained herein, nor any right to reproduce or disclose any part of the contents hereof, without the prior written consent of RFMicron.

RFMicron reserves the right to make changes, at any time and without notice, to information published in this document, including, without limitation, specifications, and product descriptions. This document supersedes and replaces all information delivered prior to the publication hereof. RFMicron makes no representation or warranty, and assumes no liability, with respect to accuracy or use of such information.

Customer is solely responsible for the design and operation of its applications and products using RFMicron products. It is the Customer's sole responsibility to determine whether the RFMicron product is suitable and fit for Customer's applications and products planned, as well as for the planned application and use of Customer's end user(s). RFMicron accepts no liability whatsoever for any assistance provided to Customer at Customer's request with respect to Customer's applications or product designs. Customer is advised to provide appropriate design and operating safeguards to minimize the risks associated with its applications and products.

RFMicron makes no representation or warranty, and assumes no liability, with respect to infringement of patents and/or the rights of third parties, which may result from any assistance provided by RFMicron or from the use of RFMicron products in Customer's applications or products.

RFMicron represents and Customer acknowledges that this product is neither designed nor intended for use in life support appliances, devices, or systems where malfunction can reasonably be expected to result in personal injury.

This product is covered by U.S. patents 7586385 and 8081043; other patents pending. Chameleon® and Magnus® are trademarks of RFMicron.