Wireless Valve Position Monitoring: A MEMS Approach

Fabien Chraim, Kris Pister
Berkeley Sensor and Actuator Center, University of California, Berkeley, USA
{chraim,pister}@eecs.berkeley.edu

***Abstract*—This paper presents a Microelectromechanical systems (MEMS) solution to rotary valve position monitoring. With a device holding commercial off-the-shelf inertial sensors, we develop a sensing algorithm encompassing self-calibration, and noise and drift removal. Our repeatable results show an accuracy of $\pm 5^\circ$ for part-turn valves and $\pm 10\%$ of a turn for multi-turn models. Our small energy-efficient wireless solution is a cheap and reliable replacement for products on the market.**

I. Introduction

From controlling the flow of water, to controlling machines, valves have been with us for centuries. Monitoring the position of valves is clearly beneficial for plant operators. The capability of detecting incorrect valve positions can help avoid the types of accidents that most often lead to personnel injuries and damaged equipment. Attempts to prevent such accidents commonly involve sending employees to various parts of the plant in order to verify valve positions. However, the use of wireless communication to accomplish this task is more time and cost effective. Additionally, access to plant-wide valve position reports will aid operators in making sense of each process pressure gradients [3]–[7].

In some market reports, experts claim that up to 85% of valves are left without any monitoring. Such a low penetration rate for valve position sensing may be attributed to the fact that sensing technology has been largely stagnant for the past two decades. Most solutions employ some sort of magnetic or optical encoding while others feature contact switches [5]–[7]. This makes most products expensive, bulky, and power-inefficient, mainly because they are offered as replacement instrumented valves, as opposed to just a sensor. Additionally, the installation of these solutions is often problematic: in many cases, a complete halt of the process is required to retrofit an existing valve with new sensors. Individual sensor calibration is also a significant barrier to speedy installations. The industry as a whole seems to be focused on satisfying ISO standards and making their products explosion-proof, while paying virtually no attention to the sensing technology. The advantages of MEMS sensors with their easy packaging and their intrinsic safety appear to have completely missed this market.

In this paper, we address the problem of valve position monitoring through the use of MEMS sensors. A cheap "peel and stick" wireless solution is presented that requires no calibration and consumes very little energy. Our main contribution is in addressing the issues described above: cost, calibration, energy consumption, size and ease of installation. Along with proposing the hardware, we present a sensing algorithm described in section IV before concluding with our results in section V.

II. Experimental Setup

This study is focused on rotary valves. In particular, we look at multi-turn valves and quarter-turn ones (though the same work can be extended to any part-turn model). Seen in figure 1 is our experimental setup with two valve models: a gate valve and a ball valve. The goal is to develop a way to keep track of the number of turns on a multi-turn valve by using the gate model, while the ball valve will be used to develop the quarter-turn angle tracking. Also seen in the figure is GINA (general inertial and navigation assistant) [8], the sensing platform of choice. This board holds the MSP430f2618 from Texas Instruments, along with a 9-axis Inertial Measurement Unit (see table I). Other than the sensors, the microcontroller interfaces with the WirelessHART compliant DN2510 radio by Dust Networks [9].

With the setup in place, we ran several experiments where data from all the sensors was collected and transmitted to a basestation. The first step in this study was to select the sampling frequency of the sensors. It is desirable in this case to lower this frequency as much as possible since each sample requires on-board communication with the sensor, some sort of analog-to-digital conversion, and processing at the microcontroller. Initially, the data was collected at 300 Hz. However, after looking at the frequency content of this data, we noted that a sampling frequency of 17 Hz was enough to capture all the motion information. All subsequent experiments were therefore run at that frequency. In each of the experiments, the valves were turned from one extreme to the other repeatedly, and in different valve orientations (horizontal, vertical and oblique).

III. Detection with Noisy Sensors

Tracking the valve angle in its plane of rotation is not as straightforward as one may think. Indeed, MEMS sensors are generally noisy or prone to drifting. While it is tempting to task this problem to the processing power of a PC, this is impossible given that the detection needs to happen on board. Consequently, all the valve angle estimations will be executed on simple 8 or 16 bit microcontroller. Another possibility would be to transmit all the sensor data to a server for

Type	Manufacturer	Part Number	Features	Active Power
Microcontroller	Texas Instruments	MSP430F2618	16-bit, 16MHz, 116kB flash, 8kB RAM	27 mW (@16MHz)
3-axis accelerometer (sensitive)	STMicroelectronics	LIS344ALHTR	+/-2 Gs or +/-6 Gs, 1.8 kHz, 660 mV/G, 50 uG/rtHz	2.1mW
3-axis accelerometer (large range)	Kionix	KXSD9-1026	+/-8 Gs, 2 kHz	0.66mW
3-axis gyroscope (with Temperature)	Invensense	ITG3200	2000 degs/s	19.5mW
3-axis magnetometer	Honeywell	HMC5843	+/- 6 Oe, 116Hz	2.7mW

TABLE I
GINA SENSING PLATFORM HARDWARE DESCRIPTION

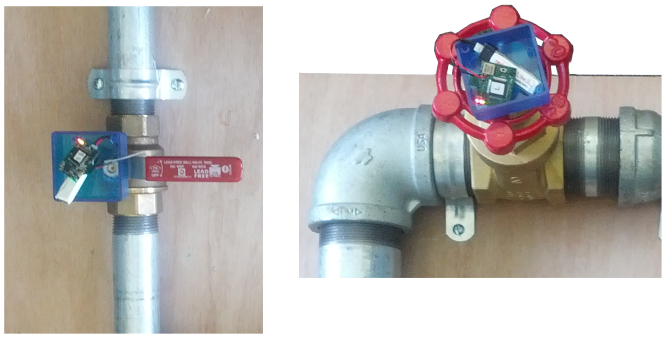

Fig. 1. The experimental setup showing an instrumented ball valve (left) and an instrumented gate valve (right)

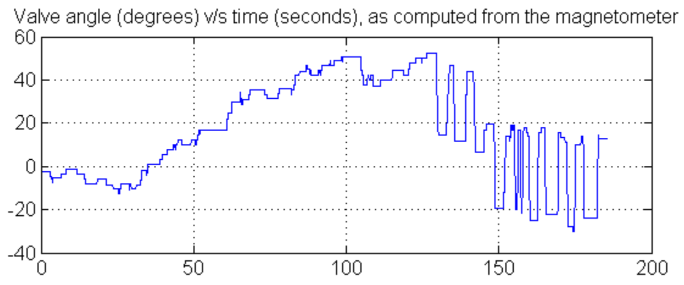

Fig. 2. Plot of the valve angle over time, as inferred from the magnetometer. The valve was steady for the first 120 seconds (no motion), then it was turned from one end to the other repeatedly. Clearly the magnetometer recorded significant motion when there was none, then showed a repetitive swing of around 40° instead of 90°

processing. However, as sending this data is likely to consume more charge than processing it locally, this would result either in congestion at the network level, or a misuse of the energy reserve.

The initial plan was to use the magnetometer to compute the orientation of the sensor. However, collecting preliminary data in the vicinity of the pipes revealed that the magnetic field around them is highly non-uniform. Additionally, considering the environment where the devices will ultimately be placed (factories and plants), the presence of metallic machines and large moving vehicles will make the magnetic environment even more unpredictable. Figure 2 shows the results of an experiment run on the ball valve. In this time series, the valve angle is computed from the magnetometer data and plotted over time. Although the first 120 seconds were supposed to show an idle device, the magnetometer picked up significant variations. During the motion of the ball valve between the two extremes (a range of $90°$), the magnetometer picked up a reduced range (about $40°$) while constantly drifting.

The magnetometer was ultimately ruled out from the list of possible sensors. Instead, we will present how the gyroscope and accelerometer were used to figure out the valve angle. There are two types of drift in MEMS gyroscopes. One of them is due to the temperature of the sensor, while the other is a result of manufacturing constraints. MEMS accelerometers suffer from noise issues. Next, we will see how each of these issues was tackled.

A. Gyroscope Temperature Self-Calibration

MEMS Gyroscopes are inherently sensitive to temperature. Luckily, this relationship is linear (in our device) over a wide range, meaning that one can easily calibrate each axis with respect to the on-board temperature reading of the chip. In our study, we imposed the constraint that each sensor cannot be calibrated independently before installation. Instead a method needed to be devised to allow the sensor to self-calibrate. When the sensor is first turned on, the operating of current heats the device. Keeping the sensing platform stationary in this initial step allows for the self-calibration of the gyroscope. We took advantage of the initial heating of the stationary device to establish the compensation function. Seen in figure 3 is the plot of the Z axis reading versus temperature for the first minute of operation, where the sensor is sampled every 3 ms. From this data, one can easily derive the slope and intercept of this line, through an Ordinary Least Square approach.

Given a vector of n temperature readings, padded with a column of ones and a vector of sensor data (say the X axis of the gyroscope when it is stationary),

$$A_{n,2} = \begin{pmatrix} T_1 & 1 \\ T_2 & 1 \\ \vdots & \vdots \\ T_n & 1 \end{pmatrix}, B_{2,1} = \begin{pmatrix} slope \\ intercept \end{pmatrix}, \hat{Gx}_{n,1} = \begin{pmatrix} \hat{gx}_1 \\ \hat{gx}_2 \\ \vdots \\ \hat{gx}_n \end{pmatrix} \quad (1)$$

here is the method used in order to find the slope and intercept of the linear relationship.

$$AB = \hat{Gx} \quad (2)$$

$$A^TAB = A^T\hat{Gx} \quad (3)$$

$$B = (A^TA)^{-1}A^T\hat{Gx} \quad (4)$$

The result is then used in the following fashion, where Gx

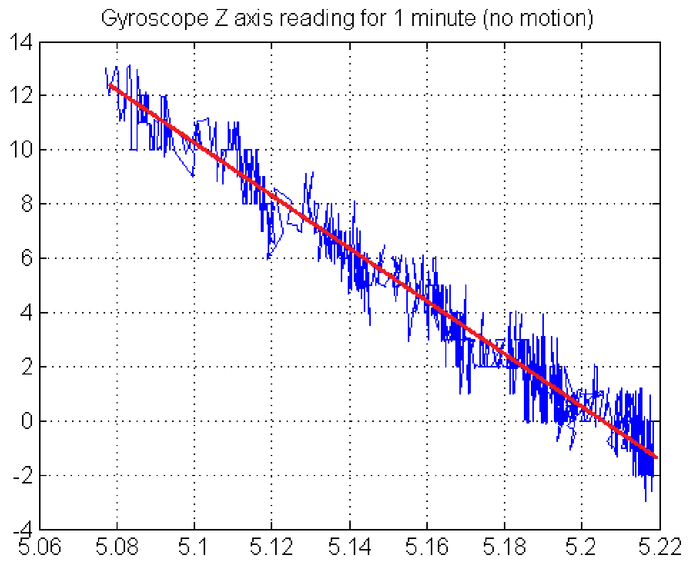

Fig. 3. Gyroscope Z axis reading versus time for temperature calibration. The plot shows the linear dependency of the gyroscope reading on temperature. It is to note that the temperature values are in arbitrary units

is the temperature compensated gyroscope vector. The same procedure is repeated for all three axes.

$$Gx = \hat{Gx} - AB \tag{5}$$

B. Gyroscope Motion Drift

A calibrated gyroscope yields the angular rate of a rigid body (valve) along three axes, in the body reference frame. In order to integrate those rates and find the valve angle, one must first rotate the rates to a fixed reference frame. This mapping is however not unique and could potentially result in a loss of information (known as the *gimbal lock problem* [2]). The solution we used involves the introduction of quaternion algebra, an extension of complex number algebra.

MEMS gyroscopes drift, mainly due to manufacturing constraints (this drift is not related to temperature). This can be seen when integrating the angular rates obtained from a sensor rotating back and forth between two positions. The resulting angle is not equal to zero as one would expect. Instead, a DC bias appears and varies over time. In our experiment in particular, when looking at the ball valve, one can see that moving the valve away from its original position and back results in a distancing from the original angle (refer to figure 4). Since there was no way of removing the drift from the gyroscope itself, we looked at the accelerometer data. By capturing the distribution of the gravity vector along the three axes when the valve is in its original position, we were able to "zero" the angle recorded whenever we observed the same distribution on the accelerometer. To ameliorate the signal-to-noise ratio on this sensor, the platform was mounted in such a way to have a significant proportion of the gravity vector on each of the axes. Of course, we needed to detect when the valve was stationary to perform this correction as, when the valve moves, the accelerometer also picks up this motion. We will look at the results of this operation in section V.

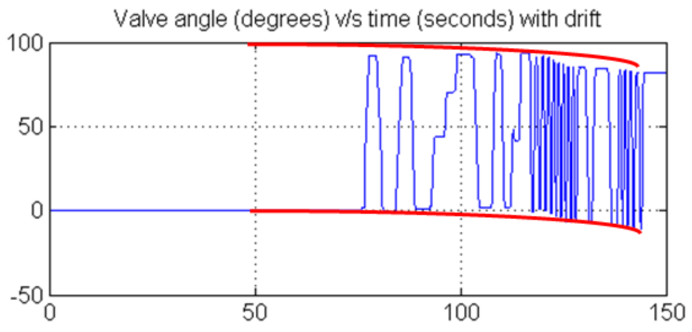

Fig. 4. Integrating the gyroscope data yields a significant drift. This drift occurs only during motion, as the valve is moved from one end to the other ($0°$ to $90°$)

C. Filtering Accelerometer Noise

Accelerometers do not drift as significantly as gyroscopes. However, they suffer from noise, even when stationary. This can be seen in figure 5 which tracks the motion on each of three accelerometer axes as the ball valve moves. In this study, since we only require the accelerometer to capture the stationary gravity vector and not the motion of the valve, only smoothing was necessary. As such, a 1st degree Robust Locally Weighted Regression filter was used [1], with a window size of 20 samples. This LOWESS filter turned out to be much better than a simple moving average filter in getting rid of erratic accelerometer noise peaks. Though computationally more intensive compared to the rolling mean approach, the LOWESS filter was necessary in getting rid of those outliers, and yielded superior smoothing with a quicker settling time. Finally, seen in figure 6 is the time series representation of the same three axes, but at the output of the filter. We note that the first few seconds of accelerometer inactivity are characterized by a settling of the internal Infinite Impulse Response (IIR) filter on each axis, which explains the slight increase or decrease of sensor reading during that time. Clearly, looking at the difference between figures 5 and 6, we can see that the signal to noise ratio has decreased significantly, allowing us to use the accelerometer reading to extrapolate as to whether the valve has returned to its original position or not, as it moves between the two extremes.

IV. Algorithm

The detailed pesudocode for the detection algorithm used to compute and report the valve angle is shown in Algorithm 1.

The function *initialize()* is implemented as defined in the above section, to incorporate the self-calibration. The *rotate()* function moves the gyroscope data from the rigid body reference frame to the Earth reference frame. The *filter()* function takes acceleromter data and returns a filtered version, as described in the part concerning the LOWESS filter. Finally, the function called *wireless.send()* transmits the data over the WirelessHART network.

Algorithm 1 Valve position monitoring algorithm

```
1: $initialize()$
2: $state \leftarrow IDLE$;
3: $valveAngle \leftarrow 0$;
4: $idleCounter \leftarrow 0$;
5: while $TRUE$ do
       $sleep(SAMPLING\_TIME)$
6:     if $state == IDLE$ then
7:         $gyroData \leftarrow gyro.sample()$
8:         if $gyro.isMoving(gyroData)$ then
9:             $state \leftarrow MOVING$
10:            $rotatedGyroData \leftarrow rotate(gyroData)$
11:            $valveAngle \leftarrow integrate(valveAngle, rotatedGyroData)$
12:        end if
13:    else if $state == MOVING$ then
14:        $gyroData \leftarrow gyro.sample()$
15:        if $gyro.isMoving(gyroData)$ then
16:            $rotatedGyroData \leftarrow rotate(gyroData)$
17:            $valveAngle \leftarrow integrate(valveAngle, rotatedGyroData)$
18:        else if $++idleCounter \bmod SAMPLING\_TIME == 0$ then
19:            if $abs(valveAngle) \leq 10°$ then
20:                $accelData \leftarrow accel.sample()$
21:                $filteredAccelData \leftarrow filter(accelData)$
22:            end if
23:        else
24:            if $abs(valveAngle) \leq 10° \& filteredAccelData == originalAccelData \pm \sigma_{noise}$ then
25:                $valveAngle \leftarrow 0$
26:            end if
27:            $state \leftarrow IDLE$
28:            $wireless.send(valveAngle)$
29:        end if
30:    end if
31: end while
```

The sensing routine shown here is designed with energy consumption in mind. When the valve is not moving (in the *IDLE* state), only the gyroscope is sampled and a very basic compare function used to determine whether the device is in motion or not. This is commonly known as Lebesgue sampling. The set of instructions that occur during the motion of the valve are simple additions and multiplications, which do not constitute a heavy microcontroller load. The heaviest function (computationally) is the *filter()* one since it runs two iterations of additions and multiplications. We note however that this function is only called after the valve returns to the idle position and only on 20 to 40 accelerometer samples. This shows that the entire routine can run on a basic microcontroller without presenting a heavy load on the batteries, allowing for an extended device lifetime.

V. Results

In this section, we present the results following a number of experiments run as described in section II. In each of these experiments, the sensing platform is oriented on an incline with respect to the plane of motion of the valve. The valve itself is placed in various orientations, then moved from one extreme to the other repeatedly, sometimes stopping at points in between and returning to the origin. Note that the first minute of inactivity is reserved for the temperature correction routine, as explained earlier.

A. Quater-Turn Valve

Figure 7 shows the valve angle vs. time for a series of experiments. As expected, the valve angle varies between $0°$ and $(\pm)90°$ (depending on the starting angle). In all five time series, and during the first minute of the experiment, the valve angle remains at $0°$, showing that the temperature compensation is working correctly. During the remainder of the experiment, we can observe that when the angle returns to a value close to zero, and as the valve stops moving, if the accelerometer detects a return to the origin, then the valve angle is adjusted accordingly. This is most apparent after the second peak of the last time series: about a second after the valve has returned to its original angle, the accelerometer forces the angle to return to zero, resulting in a small "bump" following the peak. On the figure the '$*$' labels indicates some

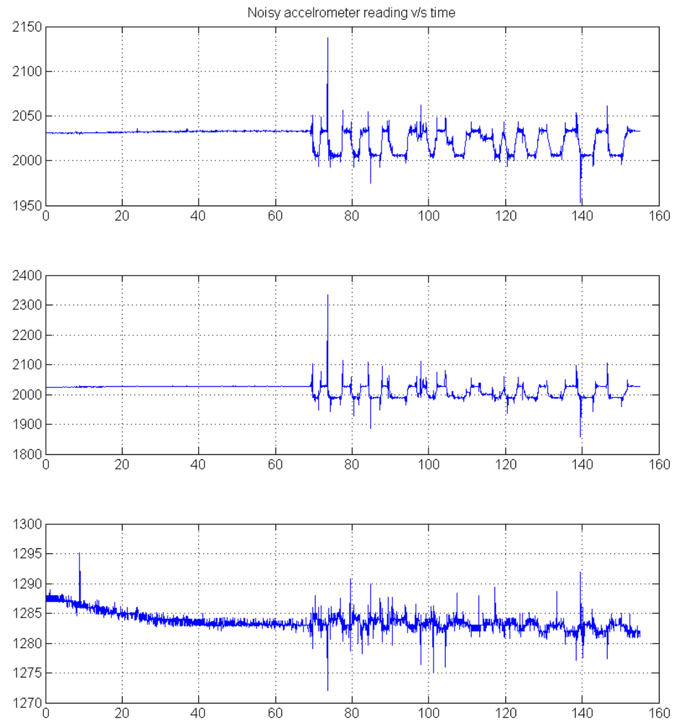

Fig. 5. Unfiltered Accelerometer readings on all three axes during the motion of the ball valve (arbitrary units). Motion starts 70 seconds in, and the valve is moved between 0° and 90° repeatedly

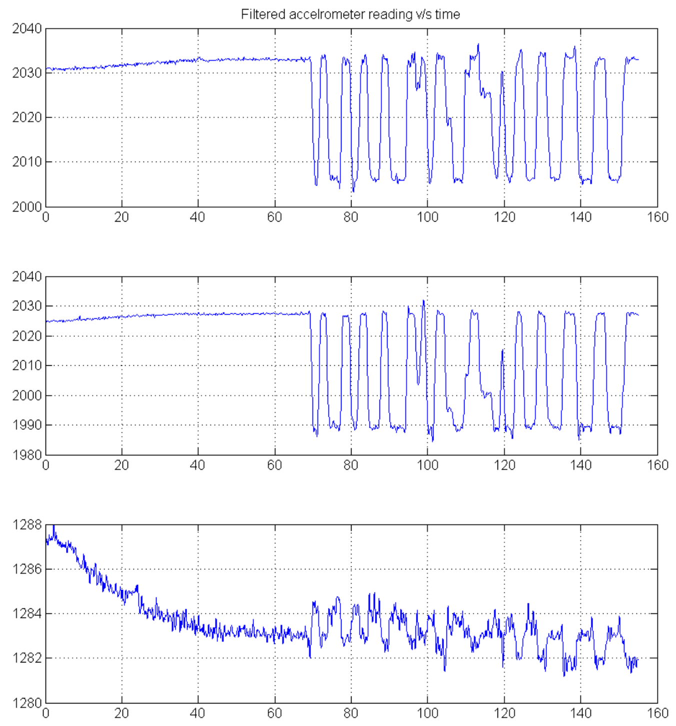

Fig. 6. Same time series as in figure 5, but at the output of the LOWESS smoothing filter

instances of using the accelerometer to correct the valve angle. The 'P' labels correspond to a partial opening or closing of the valve, while 'Q' shows an area where a quick valve movement was recorded. The same code was run on all of the time series, demonstrating that the results are repeatable and orientation independent. Comparing these results with the ground truth (measured using protractor), we recorded an accuracy of $\pm5°$. This means that our angle estimate can only be off by that amount (in one direction or the other), when the valve is away from its original position.

B. *Multi-Turn Vavle*

When presenting the results for the multi-turn valve, we decided to use a congruence modulo operator on the angle, so as to let it swing between $-180°$ and $+180°$. This is a convenience measure because it allows us to easily count the number of turns in each motion. These results can be seen in figure 8. Defining a *full cycle* as a movement from one extreme to the other and back, each of the time series depicts two full cycles (8 full revolutions). As with the quarter-turn valve, the temperature compensation is obviously working, and the gyroscope only drifts when in motion. Again, the accelerometer is used to "zero" the angle when the resting position is reached. By design, the valve we installed in our experimental setup has both extreme positions pointing in the same direction. This is apparent in time series where both *fully open* and *fully closed* positions are zeroed. However, when one tightly closes the valve (as can be seen in the second cycle of the third and fifth series), the resulting heading is slightly away from the open position, which justifies the distancing from the original angle that is observed in some experiments. On the figure, a '$*$' represents places where the valve returned to its original angle and the accelerometer data used to set the measured angle to zero. A 'T' shows when the valve was tightly closed and the angle therefore deviated from zero. As with the ball valve, the results were repeatable, yielding an accuracy of $\pm10\%$ of a turn (or about $\pm1\%$ of the entire range of the valve).

C. *System Architecture*

Our results show that with our algorithm and sensing platform, we can reliably determine the valve angle with no prior knowledge of the valve orientation or plane of motion, and with no calibration required. In our experiments however, the valve was always in either "fully-open" or "fully-closed" positions. We cannot assume that this will be the case for real-world installations. In fact, in many situations, it is undesirable for the plant operators to change the position of the valve in order to install a new sensor. This means that the valve can be partially open or closed when the platform is installed. A problem then arises of keeping track of the valve position using only relative angles. This can be easily solved however by tracking the evolution of the angle at the server side. As the valve is moved by plant employees, the sensing platform itself does not need to be aware of the position of the valve. The server-side application (or *Plant Management System*) will be

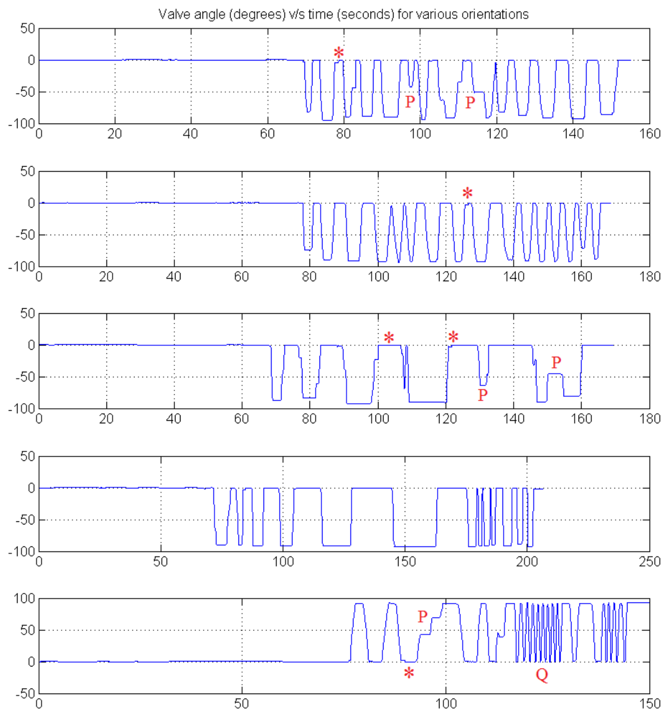

Fig. 7. Results of five experiments run with the ball valve. In each time series, the first minute (approximately) of inactivity is reserved for temperature calibration. Then the valve is moved from one end to the other, with varying motion types. A '$*$' represents the places at which the accelerometer was used to remove the gyroscope drift.

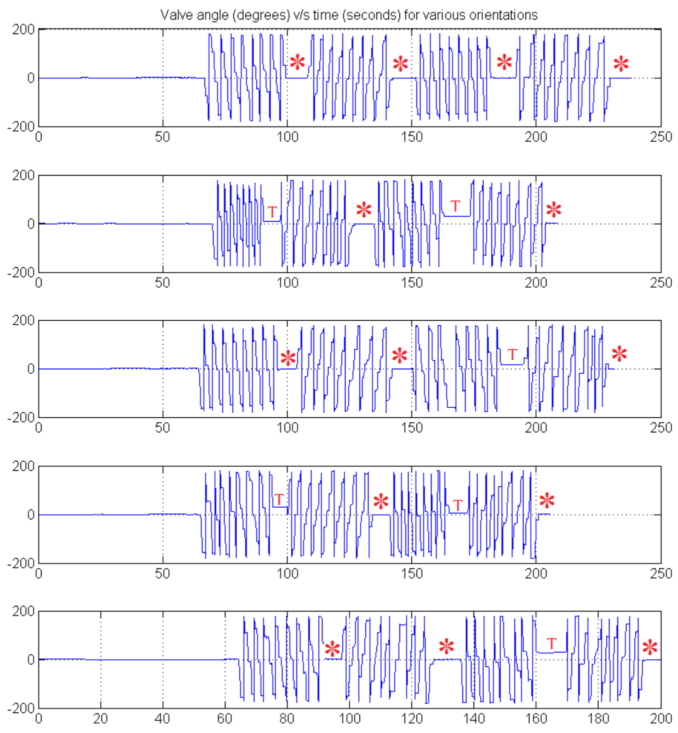

Fig. 8. Results of five experiments run with the gate valve. In each time series, the valve is closed and opened twice (8 full turns each way). The '$*$' sign shows alignment with the origin and a zeroing of the angle, while a 'T' marks a tight close.

responsible for figuring out this absolute position and raising alarms in case this position is problematic. Additionally, the algorithm can be augmented with a feature to dynamically adjust its saved accelerometer values that are used to "zero" the gyroscope drift. Another alternative would be to enable the server application to directly contact the valve and inform it of its absolute position.

VI. Conclusion

In this paper, we have presented a "peel and stick" solution to the valve position monitoring problem. Our MEMS based sensing device will ultimately be inexpensive to manufacture, as it uses sensors similar to the ones found in mobile phones and other consumer electronics products. In its present form, the board is already small enough for the majority of valves but can be made even smaller. Additionally, by employing our simple algorithm, our solution is calibration-free and energy-efficient. Using WirelessHART to report changes in position, we estimate the battery life between 5 and 10 years, depending on battery size and how often the valve is used. A valve angle accuracy of $\pm 5°$ was recorded for quarter-turn models, while an accuracy of $\pm 10\%$ of a turn (or about $\pm 1\%$ of the entire range) was obtained with the multi-turn valves. Obviously, it is easy to retrofit existing part-turn and multi-turn valves with our device. However, it is equally simple to integrate with new valve designs. Our accurate angle measurements demonstrate a reliable replacement for today's aging valve monitoring solutions.

VII. Acknowledgements

We would like to thank Chevron ETC and Chevron ITC for funding and supporting this project.

References

[1] Cleveland, William S. "Robust locally weighted regression and smoothing scatterplots," Journal of the American statistical association 74 no. 368 (1979): 829-836.

[2] Favre, J., B. M. Jolles, O. Siegrist, and K. Aminian. "Quaternion-based fusion of gyroscopes and accelerometers to improve 3D angle measurement." Electronics Letters 42, no. 11 (2006): 612-614.

[3] Hebert, D., ed. "Wireless Tech for Remote Valve Monitoring," Control 24 no. 11 (2011): 47.

[4] Kurtis Jensen, "The Reliability and Security of Wireless Valve Monitoring," Valve Magazine 20, no. 3, Summer 2008.

[5] Westlock Controls Wireless Valve Monitoring System, 2013, Westlock Controls. 13 March 2013, http://westlockcontrols.com/Images/WESTDS-09000-US-1205.pdf

[6] Radomsky, Israel, Reuben Fuchs, Israel Kalman, Amir Nemenoff, and Ohad Gal. "Device and system for monitoring valves." U.S. Patent 7,886,766, issued February 15, 2011.

[7] Honeywell OneWireless XYR 6000 Valve Position Sensor, 2013, Honeywell. 13 March 2013, http://sensing.honeywell.com/honeywell-sensing-xyr6000-valve-position-sensor-productsheet.pdf

[8] Ankur M. Mehta; Kristofer S. J. Pister; "WARPWING: A complete open source control platform for miniature robots," 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Taipei, Taiwan, Oct. 2010.

[9] HCF - HART Communication Foundation, "HART7 Specification", September 2007