APPARATUS AND DEMONSTRATION NOTES

The downloaded PDF for any Note in this section contains all the Notes in this section.

Frank L. H. Wolfs, *Editor*

Department of Physics and Astronomy, University of Rochester, Rochester, New York 14627

This department welcomes brief communications reporting new demonstrations, laboratory equipment, techniques, or materials of interest to teachers of physics. Notes on new applications of older apparatus, measurements supplementing data supplied by manufacturers, information which, while not new, is not generally known, procurement information, and news about apparatus under development may be suitable for publication in this section. Neither the *American Journal of Physics* nor the Editors assume responsibility for the correctness of the information presented.

Manuscripts should be submitted using the web-based system that can be accessed via the *American Journal of Physics* home page, http://www.kzoo.edu/ajp/ and will be forwarded to the ADN editor for consideration.

Using XBee transducers for wireless data collection

Eric Ayars[a)]

Department of Physics, California State University, Chico, California 95929-0202

Estella Lai

Pleasant Valley High School, Chico, California 95926

(Received 20 July 2009; accepted 19 April 2010)

This article describes how to use XBee transducers to create small and lightweight wireless sensors, which send data to a base station for collection and analysis. Data collection is limited to 10-bit accuracy by the XBee hardware. Depending on the type of XBee used, up to six data channels can be transmitted over a range of up to 15 miles. We describe the technical details of the process using the low-power version of the XBee transducer and a three-axis accelerometer chip. © *2010 American Association of Physics Teachers.*

[DOI: 10.1119/1.3427415]

I. INTRODUCTION

Physics teachers today have an amazing array of sensors available for use in teaching laboratories. There are individual chips that can measure capacitance, temperature, humidity, barometric pressure, GPS coordinates, magnetic field, orientation in space, and just about anything else we could want.

One type of chip that is of particular interest is the microelectromechanical system (MEMS) accelerometer chip. This chip uses the deflection of a microscopic silicon cantilever to measure acceleration. These chips can cover a range of accelerations from $\pm 2g$ to $\pm 18g$ with milli-g (≈ 0.01 m/s^2) precision. Many chips simultaneously measure the acceleration along three independent axes. The MEMS chips are used in applications such as automotive safety systems, where they trigger airbags when the acceleration exceeds a certain limit or call emergency responders automatically if they sense that the vehicle is upside down. They are also used in consumer electronics to change the screen orientation of smartphones or park hard-drive heads safely away from the disk surface if they detect that a laptop is in free-fall. Because of these high-volume applications, MEMS accelerometer chips have become surprisingly inexpensive.

To use any of these sensors requires a way to get the measurement from the chip to a computer. Wires typically work well for this purpose, but there are some measurements—acceleration in particular—which can be adversely affected by these wires. The XBee radio transducer[1] provides a solution to this problem. XBee transducers use the 2.4 GHz band for either point-to-point or network communications. The simplest low-power version of the XBee has a line-of-sight range of about 100 m. The XBee is also available in high-power versions with antenna options that can boost their useful range up to 15 miles. The XBee comes standard with six 10-bit analog-to-digital converters and can thus convert the analog outputs of multiple sensors directly to digital format prior to sending the information to the computer.[2]

In the simplest configuration, one XBee ("local") is connected via a USB/serial adaptor to the data-collection computer and a second XBee ("remote") is connected to the sensor(s). The remote XBee is configured to sample the voltage at the desired channel(s) at a specific rate and send the measurements to the local XBee. The data are then read from the local XBee by the computer.

II. CONSTRUCTING THE REMOTE UNIT

The remote unit requires three elements: A power supply, one or more sensors, and an XBee.

The XBee requires a regulated 3.3 V supply. In order to make the remote unit small and light, we used a single 1.5V N-cell battery and a voltage up-conversion circuit[3] as our power supply. Although the up-converter increases the inefficiency of the power source, we found that this supply was

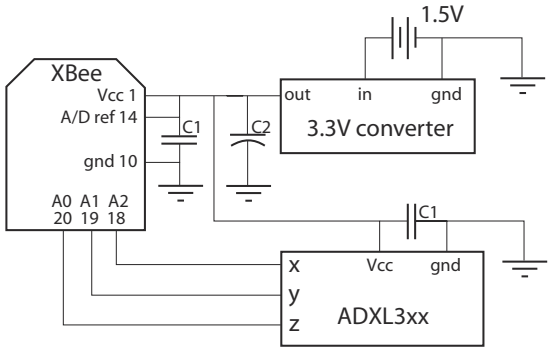

Fig. 1. Circuit diagram for the remote acceleration sensor.

able to power the remote for at least 30 min. A larger battery could of course power the remote for much longer but would increases its mass.

Our XBee was connected to a three-axis accelerometer from the ADXL3xx series by Analog Devices. These particular chips are extremely small surface-mount devices and require several external components. We purchased the chips on preassembled breakout boards,[4] requiring us to merely connect power, ground, and three signal lines for the acceleration measurements. Since the XBee has a nonstandard pin spacing, we used an XBee adapter board[5] to interface with our standard protoboard.

We found that the noise level in our measurements decreased slightly with the addition of 0.1 μF ceramic capacitors at the V_{cc} terminals of the XBee and the accelerometer and a 47 μF tantalum capacitor at the output of the voltage up-converter. Our circuit diagram is shown in Fig. 1, and the completed remote is pictured in Fig. 2. The mass of the completed circuit with battery was 27 g. This mass could be further decreased by about 8 g if we built a single surface-mount board rather than using breakout boards. The use of a CR2032 calculator battery instead of an N-cell battery would also lower the mass but significantly reduce the data-collection time.

III. CONNECTING THE LOCAL UNIT

The XBee transducer communicates via a serial connection. Few modern computers have a serial port, but it's possible to use a USB port to communicate with a serial device via a USB/serial converter chip. We used a combined XBee breakout board and USB/serial converter[6] from Sparkfun-.com. Installation of the driver[7] for the USB/serial chip on our computer was necessary in order to communicate with the converter. This driver is available for Windows, Macintosh, and Linux and creates a virtual com port on the computer through which serial communications can occur. Communications through this virtual com port is managed via a terminal-emulation program such as ZTERM for Macintosh and Windows or SCREEN for Linux. The XBee uses an AT command set, similar to the commands used to configure modems several decades ago.

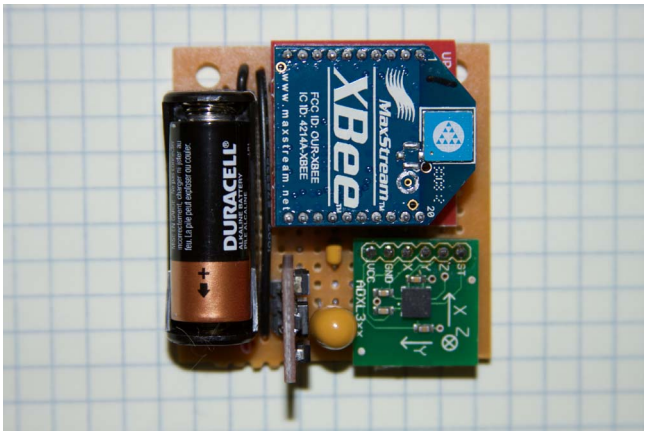

Fig. 2. Completed remote acceleration sensor, shown on a 5 mm grid.

IV. CONFIGURING THE XBEE TRANSDUCERS

We found that the XBee transducers were delivered with outdated firmware. The firmware had to be updated using the "XCTU" program[8] from Digi, the manufacturer of the XBee transducer. The XCTU program is Windows-only but runs well under PARALLELS DESKTOP on Macintosh and under WINE in Linux. XCTU is a self-explanatory program. We connected each XBee to the computer using the XBee breakout/serial converter board, then ran XCTU, and updated the XBee firmware to the newest version on *both* transducers.

Once the firmware was up-to-date, the local XBee was configured to serve as a base station. After establishing communications with the XBee transducer via our terminal-emulation program and XBee breakout/serial converter board, the network ID was set using the command "ATID*xxxx*," where *xxxx* was a four-digit hexadecimal number. The exact ID is not important unless multiple setups are communicating with different base stations identified by different network IDs. The base station address was set to 0 using the command "ATMY0." The default baud rate for serial communications with the XBee is 9600. Higher speeds are better: We found that a 115,200 baud rate worked well and configured our XBee to that rate using the "ATBD7" command. The write command, "ATWR," can be used to write these changes to the nonvolatile memory of the XBee. At this point, we marked the local XBee with a permanent marker so that we didn't mix up the transducers.

Next, the remote XBee was configured to collect the desired data and send it to the base station. This required us to connect the remote XBee directly to the computer and use the procedure described above to set the network ID, the remote address, and the baud rate. The network ID was set to that of the base station we wished the remote to communicate with, and the remote address was set to a unique value.[9] The remote XBee also must know the address of the base station: This was achieved by using the command "ATDL*x*," where *x* is the address of the base station. The command "ATD02, D12, D22" was used to configure inputs 0, 1, and 2 of the XBee as analog inputs and instruct the XBee to send the information from these inputs to the base station. The analog sampling rate was set via "ATIR*x*," where *x* was the desired interval between samples, in milliseconds. The command "ATIT1" was used to set the number of samples per packet sent back to the base station to one. The changes were written to nonvolatile memory with the ATWR command.

V. SOFTWARE

Once the remote XBee has been configured and powered, it endlessly sends packets of data to the local XBee at the

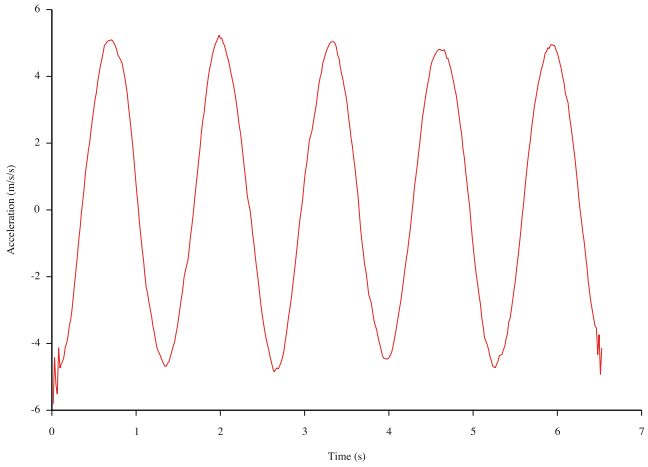

Fig. 3. Simple harmonic motion, as measured by the remote acceleration sensor. The period and amplitude both match the values obtained by measurement with stopwatch and meter stick.

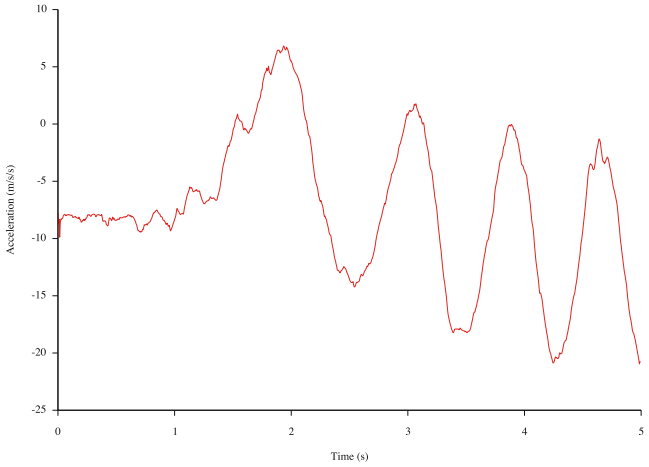

Fig. 5. Radial acceleration of a point on the inside of a bicycle rim, as the bicycle is started from rest. The initial acceleration is not 9.8 m/s^2, since the radial vector being measured was not initially vertical.

rate specified with the "ATIR" command. The computer to which the local XBee is connected must run a program that reads the packets from the local XBee, extracts the A/D values, converts them to accelerations, and saves and/or graphs the values. The packets are sent in binary format, so whatever language used to read the packets must include a library for conversion to human-readable format. We wrote our program in Python using the XBEE.PY (Ref. 10) package to parse the incoming packets, but libraries for the XBee are available in many other languages.

The data sent by the XBee are in the form of 10-bit integers. The analog-to-digital converter of the XBee compares the input voltage at its analog input with the reference voltage connected to Pin 14 (see Fig. 1) and returns N where $N/2^{10}$ is the approximate ratio of the input voltage to the reference voltage. The reference voltage in our case is 3.3 V, and an input of 3.3 V on an analog line would result in a transmitted value of $2^{10}=1024$.

We calibrated the XBee by observing the average values returned for two known accelerations: When the remote was at rest on the floor face-up (N_{up}, $a=g$) and when it was face-down (N_{down}, $a=-g$). Since the output of the A/D converter is linear, we used a linear calibration function passing through the points (N_{up},g) and $(N_{\mathrm{down}},-g)$,

$$a(N)=-g\frac{(N_{\mathrm{up}}+N_{\mathrm{down}})}{(N_{\mathrm{up}}-N_{\mathrm{down}})}+\left[\frac{2g}{(N_{\mathrm{up}}-N_{\mathrm{down}})}\right]N. \tag{1}$$

When we worked purely in the vertical direction, we chose to subtract the "rest acceleration" g from the result of Eq. (1). This rest acceleration arises because the accelerometer determines acceleration by measuring the deflection of an internal silicon cantilever. When the accelerometer is at rest in a gravitational field g, the deflection of the cantilever is the same as when the accelerometer is accelerating at g in free space.

The time information associated with the packets transmitted by the XBee was a problem. Although the XBee was configured to send packets at set time intervals, it did not do so. The packet interval was proportional to the set interval and was apparently regular, but the actual interval was not the interval specified by the "ATIR" command during the

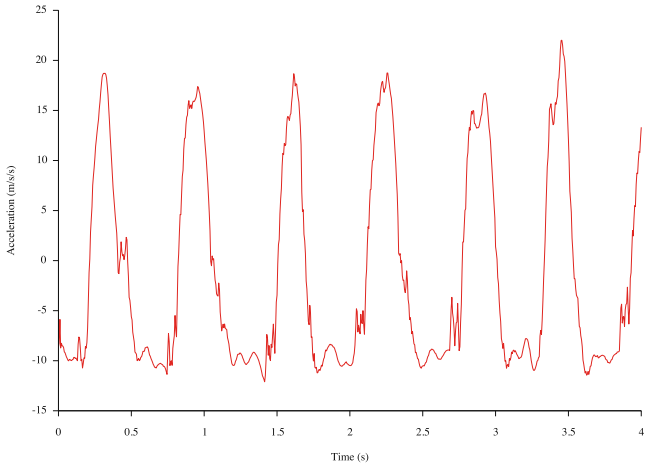

Fig. 4. Acceleration measured at the frame of a pogo-stick in use. One can see the peak acceleration of the frame, the periods of free-fall, and the impacts of the pogo-stick piston against the ground and the frame stop at the beginning and end of each ground-contact period.

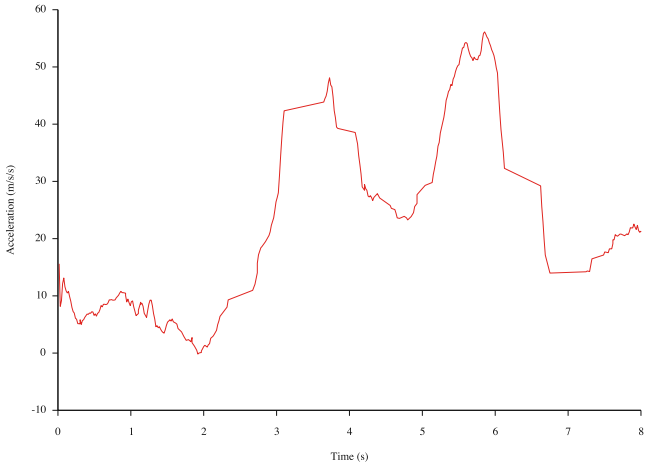

Fig. 6. The z component of the acceleration of a radio-controlled airplane during a loop, as measured in the plane's frame of reference.

configuration of the XBee. We worked around this problem by having the computer record the time at which each packet was received. These times were not absolute due to unknown transmission delays, but the time intervals appeared to be accurate to within our measurement uncertainty. Since we cared more about the time intervals between various events than the exact time of the events, this solution was acceptable.

A sample program that collects data from one acceleration axis for a user-defined time interval is provided on our website.[10]

VI. SAMPLE RESULTS

Simple harmonic motion served as the first test of our wireless accelerometer. This test allowed us to compare the signal from the accelerometer with a well-characterized motion. We attached the sensor assembly to a mass at the end of a spring and compared the period of the data with the observed period of the motion of the mass. There was more noise in the acceleration data than we expected, but with minimal digital filtering the results were quite usable and closely matched expected values, as shown in Fig. 3. Note that the rest acceleration has been subtracted from the values shown in Fig. 3.

Other systems that are more difficult to measure with a standard "wired" sensor include a pogo-stick (see Fig. 4), the rim of a bicycle tire (see Fig. 5), and a radio-controlled airplane performing a loop (see Fig. 6). For the pogo-stick graph, the rest acceleration was subtracted. For the other two examples it was not meaningful to do this as the direction of the acceleration was neither vertical nor constant.

The data from the radio-controlled plane was particularly noisy due to vibration from the engine. There were also several stretches where the data dropped out possibly due to the distance between the plane and our receiver or electrical interference from either the motor or the R/C transmitter. Nevertheless, Fig. 6 is particularly interesting from a teaching perspective as the accelerometer measures only the magnitude of the z component of the acceleration, not the direction. The direction of the plane's z axis changes by 360° during the course of the loop. The plane is initially in approximately level flight ($a_z \approx 9.8\ \mathrm{m/s^2}$). The z component of the acceleration increases sharply as the plane enters the loop, decreases as the plane is inverted at the top of the loop, and then increases sharply again as the plane pulls out into level flight once more. Also note the magnitude of the acceleration: In another flight we measured an acceleration of over $75\ \mathrm{m/s^2}$!

VII. CONCLUSIONS

Although there are significant technical details that must be managed when setting up XBee transducers as wireless sensors, the technique shows great promise in facilitating measurements that would not otherwise be possible.

One limitation of the XBee transducers is the 10-bit A/D converters. This limits the resolution of our measurements to one part in 1024, which can present a problem if one wishes to make high-resolution measurements with a large dynamic range.

We limited our applications to measurements of acceleration in this initial exploration of wireless sensing. The XBee is in no way limited to acceleration measurements though! It can measure *anything* that can be expressed in terms of a 0–3.3 V signal.

More details about this technique are available on the authors' web pages at http://phys.csuchico.edu/ayars/XBee/welcome.html.

ACKNOWLEDGMENTS

The authors would like to thank Tom Igoe, whose excellent books and articles provided both impetus and guidance for this exploration. The authors would also like to thank J. P. Wooley for spending a morning flying our sensor around in his radio-controlled planes.

[a]Electronic mail: ayars@mailaps.org

[1]Available from Sparkfun.com, http://www.sparkfun.com/commerce/product_info.php?products_id=8664. Note that there are two versions of the XBee hardware: The original (often called "series 1") and the newer "series 2.5." The two series are not interchangeable. Series 2.5 has better mesh networking capabilities but will only accept A/D inputs of up to 1.2 V. Series 1 is preferred for this application as it is somewhat easier to set up point-to-point communications and will accept A/D inputs of up to the 3.3 V power supply voltage.

[2]More information about the use of XBee transducers can be found in Tom Igoe, *Making Things Talk: Practical Methods for Connecting Physical Objects* (O'Reilly Media, Cambridge, MA, 2007), pp. 222–259; and Tom Igoe, "Wireless motion sensing made easy," Make Magazine **14**, 125–128 (2008).

[3]Available from Sparkfun.com, http://www.sparkfun.com/commerce/product_info.php?products_id=9216.

[4]Sparkfun, http://www.sparkfun.com/commerce/product_info.php?products_id=847, for example. Other ranges of accelerometer breakout boards are available as part numbers SEN-00848 and SEN-00849.

[5]See: http://www.sparkfun.com/commerce/product_info.php?products_id=8276.

[6]See: http://www.sparkfun.com/commerce/product_info.php?products_id=8687.

[7]See: http://www.ftdichip.com/Drivers/VCP.htm.

[8]See: http://www.digi.com/support/productdetl.jsp?pid=3352&osvid=57&tp=4&s=316.

[9]Note that it is possible to have multiple sensors communicate with one base station simultaneously. Just make sure that each XBee on a given network ID has a unique address.

[10]See supplementary material at http://dx.doi.org/10.1119/1.3427415 for the XBee and one-channel python codes.